CC-Eval: A Benchmark for PhD-Level Mathematical Reasoning in Communication Complexity

Anonymous ACL submission

Abstract

Large language models have recently shown promising abilities in mathematical reasoning, yet their performance on foundational graduate-level theory remains insufficiently understood. In this work, we present a benchmark for assessing frontier LLMs on Communication Complexity by Kushilevitz and Nisan, a foundational textbook in theoretical computer science. We evaluate four state-of-the-art models by requiring them to generate complete LaTeX proofs for a diverse set of textbook lemmas and exercises. The results reveal substantial variation in performance: the strongest model achieves an overall accuracy of 78.4 %, while the weakest reaches only 49.5 %, with other models falling between these extremes. Beyond aggregate accuracy, we conduct a qualitative analysis of the generated proofs, highlighting systematic differences in logical structure, conciseness, and failure modes. These findings indicate that while current frontier models can assist with graduate-level learning and partial formalization in communication complexity, their reliability for producing fully rigorous proofs remains uneven. This benchmark provides a focused assessment of the strengths and limitations of LLMs on core theoretical computer science tasks.

1 Introduction

The integration of Large Language Models (LLMs) into the mathematical sciences marks a pivotal shift from traditional symbolic solvers to autonomous agents capable of high-level abstraction. Recent breakthroughs have demonstrated that neural architectures can transcend simple pattern matching to discover novel structures and optimize complex theoretical bounds (Wagner, 2021; Fawzi et al., 2022; Romera-Paredes et al., 2024; Charton et al., 2024). These advancements have birthed a new paradigm of "AI-assisted discovery," where LLMs facilitate a synergistic loop of hypothesis generation, formal verification, and empirical exploration. Consequently, the frontier of mathematical research is increasingly defined by the ability of these systems to navigate intricate, multi-layered logical landscapes with minimal human intervention.

Meanwhile, large language models have made steady progress in writing, programming, and planning tasks. (Jiang et al., 2024; Lei et al., 2025). In the mathematical domain, their utility has expanded from basic syntactic error correction to the orchestration of sophisticated proof strategies. Modern frontier models are now capable of executing long-range symbolic manipulations and constructing coherent arguments, suggesting a burgeoning capacity for structured intellectual labor.

However, a critical discrepancy remains between the anecdotal success of LLMs on isolated problems and their consistent performance across cohesive theoretical frameworks. Existing benchmarks frequently rely on fragmented datasets or competition-level problems that, while difficult, lack the cumulative logic and interconnectedness of graduate-level mathematics. This leaves a fundamental question unanswered:

Does the reasoning capability of Large Language Models truly encompass the formal mathematical frameworks essential to modern theoretical research?

To probe this cognitive boundary, we introduce a rigorous evaluation framework centered on the landmark text Communication Complexity by Kushilevitz and Nisan (Kushilevitz and Nisan, 1997).

This subject serves as a uniquely demanding testbed; it requires a synthesis of combinatorial intuition, information theory, and rigorous deductive logic. Unlike typical datasets, the exercises in this curriculum demand that a model maintain a "global state" of theoretical understanding—applying foundational results such as log-rank conjectures, dis-

crepancy bounds, and the properties of monochromatic rectangles to increasingly complex scenarios.

In this work, we systematically assess four frontier models—GPT-5-Thinking, Grok-4, Gemini-3-Pro, and Claude-Sonnet-4.5—on their ability to provide end-to-end formal LaTeX proofs. Our methodology employs a strictly controlled pipeline that separates problem formalization from the reasoning process, incorporating a dual-layer verification mechanism to ensure mathematical validity. The resulting performance distribution across models and chapters is summarized in Table 1, which reports the counts of correct, failed, and incorrect solutions for each model.

Our primary contributions are summarized as follows:

- We establish a rigorous framework for autonomous LaTeX proof generation within the domain of communication complexity, complemented by a multi-dimensional evaluation framework that quantifies formal correctness and probabilistic reasoning.
- We conduct a systematic empirical investigation of frontier LLMs (GPT-5-Thinking, Gemini-3-Pro, Claude-Sonnet-4.5, and Grok-4) to evaluate their reasoning capabilities in theoretical computer science.
- Our results demonstrate that state-of-the-art models have attained a level of reliability sufficient to serve as supports for theoretical computer science.

2 Related Work

2.1 Benchmarks for Mathematical Reasoning

The integration of AI into mathematics is fundamentally transforming the approach to complex problem-solving. Recent studies (Davies et al., 2021; Douglas et al., 2022) highlight the impressive efficacy of AI systems in tackling intricate mathematical challenges. Consequently, a diverse array of benchmarks has been developed to rigorously assess models' logical reasoning and formal proof generation capabilities. Early efforts primarily focused on elementary skills, including basic arithmetic (Roy and Roth, 2015) and introductory algebra (Ling et al., 2017). MiniF2F (Zheng et al., 2022) pioneered the systematic evaluation of core mathematical competence across domains such as geometry, number theory, and combinatorics. Subsequently, the MATH dataset (Liu et al., 2024a) established a standardized protocol for high-school and undergraduate-level problems. Expanding this scope, NaturalProofs (Welleck et al., 2021), GSM8K (Cobbe et al., 2021), MathVista (Lu et al., 2023), and OlympiadBench (He et al., 2024) incorporate natural language reasoning, proof verification, and visual grounding to provide a more holistic assessment.

2.2 Benchmarks for Video Generation

As text-to-video generation advances, developing robust metrics to evaluate temporal and semantic fidelity has become a priority. FETV (Liu et al., 2023) established an early foundation by introducing a fine-grained protocol that balances prompt complexity with traditional metrics like FID (Heusel et al., 2017), FVD (Unterthiner et al., 2018), and CLIPScore (Hessel et al., 2021). The emergence of powerful diffusion-based architectures (Ho et al., 2022; Wu et al., 2023; Yang et al., 2024) has catalyzed a shift toward more nuanced evaluation dimensions. For instance, StoryBench (Bugliarello et al., 2023) focuses on narrative continuity, while subsequent frameworks scrutinize compositional generalization (Feng et al., 2024; Sun et al., 2024), temporal consistency (Liao et al., 2024; Ji et al., 2024), and physical plausibility (Meng et al., 2024; Guo et al., 2025). To address factual accuracy, T2V-WorldBench (Chen et al., 2025) evaluates models based on world-knowledge and commonsense reasoning. Comprehensive suites such as EvalCrafter (Liu et al., 2024b) and VBench (Huang et al., 2024a,b) have since emerged as industry standards by integrating multi-dimensional automated metrics with human judgment.

2.3 Benchmarks for Image Generation

The rapid evolution of text-to-image models has necessitated increasingly sophisticated benchmarks to capture the nuances of visual synthesis. Early assessments typically utilized caption-based pairs from datasets like MS COCO (Lin et al., 2014), relying on pretrained perception models to verify object presence and scene fidelity (Ramesh et al., 2022; Hu et al., 2023). With the maturation of diffusion models, DrawBench (Saharia et al., 2022), DALL-EVAL (Cho et al., 2023), and HE-T2I (Petsiuk et al., 2022) introduced more challenging prompts to test attribute binding, vi-

Chapter	Count	Claude			Gemini			GPT			Grok		
		Correct	Failed	Incorrect	Correct	Failed	Incorrect	Correct	Failed	Incorrect	Correct	Failed	Incorrect
1	9	8	0	1	8	0	1	5	0	4	6	0	3
2	10	5	0	5	7	0	3	3	0	7	4	0	6
3	14	8	0	6	10	0	4	4	0	10	5	0	9
4	32	26	0	6	30	0	2	20	0	12	27	1	4
5	5	3	0	2	4	0	1	3	0	2	2	2	1
6	11	8	0	3	10	0	1	8	0	3	7	3	1
7	4	2	0	2	3	0	1	2	0	2	1	2	1
8	5	4	0	1	3	0	2	2	0	3	2	1	2
9	3	0	0	3	1	0	2	1	0	2	1	2	0
10	5	3	0	2	3	0	2	2	0	3	2	1	2
11	3	1	0	2	2	0	1	1	0	2	2	0	1
12	8	4	1	3	5	0	3	3	0	5	3	1	4
13	2	1	0	1	1	0	1	1	0	1	1	0	1
Total	**111**	**73**	**1**	**37**	**87**	**0**	**24**	**55**	**0**	**56**	**63**	**13**	**35**
Accuracy		65.8%			78.4%			49.5%			56.8%		

Table 1: Performance comparison across different chapters. **Count** denotes the total number of questions. For each model, we report the number of **Correct**, **Failed**, and **Incorrect** answers. Models: **GPT** (GPT-5-Thinking), **Grok** (Grok-4), **Gemini** (Gemini-3-Pro), and **Claude** (Claude-Sonnet-4.5-Thinking).

sual reasoning, and social bias. Notably, recent research (Zhao et al., 2025) has focused on rectifying failure modes in counting tasks through innovative noise regulation strategies. Furthermore, T2I-CompBench (Huang et al., 2023), ConceptMix (Wu et al., 2024), and GenAI-Bench (Li et al., 2024a) emphasize the evaluation of relational structures and complex scene compositions. To bridge the gap between pixels and physical laws, Commonsense-T2I (Fu et al., 2024) and PhyBench (Meng et al., 2024) assess adherence to real-world logic, while multimodal suites like SEED-Bench (Li et al., 2024b) and MM-Vet (Yu et al., 2023) push the boundaries toward general cross-modal inference.

3 Recent Papers about Using GPT to solve math questions

Recent work has begun systematically assessing LLMs' potential in solving research-level mathematics (Feldman and Karbasi, 2025; Diez et al., 2025; Ivanisvili and Xie, 2025; Alexeev and Mixon, 2025; Jang and Ryu, 2025; Salim, 2025; Georgiev et al., 2025). These studies, covering fields from combinatorics to number theory, highlight a clear pattern: LLMs are adept at generating ideas, counterexamples, and various constructions to optimize specific parameters. However, achieving full mathematical rigor still necessitates human intervention.

Typically, these papers utilize a consistent prompting approach, providing the model with a concise problem statement to elicit proof outlines or intermediate steps. LLMs generally succeed in exploratory tasks, such as suggesting structures or formulating helpful conjectures. Nevertheless, for problems requiring long-range or delicate technical reasoning, LLMs serve better as sources of inspiration than as providers of complete, verified proofs.

Building on these insights, we review several landmark results—including recent technical reports from OpenAI and Terence Tao's documented workflows—focusing on their problem domains, prompt engineering strategies, and the subsequent verification protocols employed by the authors.

3.1 Resolving the Point Convergence of Nesterov's Method via LLM Interventions

A notable breakthrough in LLM-assisted optimization was achieved in (Jang and Ryu, 2025), where the authors addressed the long-standing problem of the point convergence of Nesterov's Accelerated Gradient (NAG) method (Nesterov, 1983). Consider the canonical convex optimization problem: $\min_{x\in\mathbb{R}^n} f(x)$, where $f : \mathbb{R}^n \to \mathbb{R}$ is L-smooth and convex. The NAG iteration is governed by:

$$x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k),$$
$$y_{k+1} = x_{k+1} + \frac{t_k - 1}{t_{k+1}}(x_{k+1} - x_k).$$

Although the $O(1/k^2)$ objective convergence rate is well-established, the convergence of the sequence $\{x_k\}$ to a specific minimizer x_∞ remained an open question until the intervention of LLMs.

In (Jang and Ryu, 2025), the authors utilized ChatGPT to bridge this gap, establishing that

$x_k \to x_\infty$ and $y_k \to x_\infty$. The discovery process involved a recursive prompting strategy: the authors "primed" the model with existing proofs of continuous-time counterparts in LaTeX and tasked it with extrapolating these logic chains to the discrete-time setting. By iteratively refining model-generated drafts and filtering out hallucinated arguments, the authors successfully synthesized a complete, human-verified proof. This workflow exemplifies the potential of LLMs to accelerate the transition from continuous-time intuition to discrete-time formalisms.

3.2 Refuting the NICD Conjecture via LLM-Guided Counterexamples

(Ivanisvili and Xie, 2025) utilizes LLMs to identify a concrete counterexample to a long-standing conjecture in Non-Interactive Correlation Distillation (NICD) (Yang, 2007) under erasure noise. The problem considers a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ and an erasure process where each bit of the input x is independently replaced by an erasure symbol with probability $p \in [0,1]$, resulting in a corrupted vector z. The objective is to maximize the correlation measure $\mathbb{E}|f(z)|$. A classical hypothesis in this domain is that the n-bit majority function, M_n, is the optimal unbiased Boolean function for any given p. Specifically, it was conjectured that:

$$M_n = \arg \max_{f:\mathbb{E}[f]=0} \mathbb{E}|f(z)|.$$

To challenge this hypothesis, the authors employed GPT-5 Pro as a heuristic engine. The prompting strategy involved defining the function domain and the expectation-based objective, then tasking the model with proposing alternative Boolean structures that might surpass the majority function's performance. Instead of requesting a full formal proof, the authors leveraged the model's ability to navigate high-dimensional discrete spaces and suggest candidate functions.

Through this interactive process, GPT-5 Pro identified a specific 5-bit Boolean function f_5 that satisfies:

$$\mathbb{E}|f_5(z)| > \mathbb{E}|M_5(z)| \quad \text{for } p = 0.40.$$

This finding provides a definitive counterexample to the NICD conjecture for the specified parameters. Notably, while the candidate was surfaced by the LLM, the authors conducted independent manual verification to ensure the mathematical validity of the result, confirming the expectations without computational reliance.

3.3 Probing Submodular Maximization Conjectures via LLM Assistance

(Feldman and Karbasi, 2025) assesses the capability of Large Language Models to navigate the landscape of combinatorial optimization, specifically targeting open questions in submodular maximization (Nemhauser et al., 1978). The study focuses on optimization objectives of the form $\max_{S \in \mathcal{C}} f(S)$, where the set function f exhibits the diminishing returns property—formally defined by the submodularity condition:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B).$$

The scope extends to relaxed settings, such as γ-weakly submodular functions ($\gamma \in [0,1]$), under various constraint families $\mathcal{C}$ (e.g., cardinality or matroid constraints).

To evaluate model performance, the authors adopted a minimalist prompting strategy: GPT-5 was presented with raw conjecture statements devoid of auxiliary heuristics or proof sketches. The model was tasked with a binary objective—either to synthesize a proof or to construct a counterexample. This setup necessitated a rigorous "human-in-the-loop" validation framework. While the LLM acted as a generator of preliminary logic tracks and partial derivations, the researchers assumed the role of critical adjudicators. This involved the meticulous sanitization of model outputs: filtering out hallucinations, bridging logical gaps, and independently re-deriving essential inequalities. Even in cases where the model proposed seemingly complete arguments, the authors conducted an ab initio manual verification to ensure mathematical soundness.

3.4 LLM-Aided Generation of Sidon Set Counterexamples

The work of (Alexeev and Mixon, 2025) revisits Erdős's classical conjecture (Erdős, 1975; Erdös, 1977), which posits that every finite Sidon set (Sidon, 1932) is extensible to a finite perfect difference set. Mathematically, a Sidon set A is characterized by the uniqueness of pairwise differences $a - a'$ (for distinct $a, a' \in A$). The central inquiry of this study is determining the embeddability of such sets into perfect difference sets modulo v, particularly for moduli of the form $v = p^2 + p + 1$.

The authors adopted a workflow centered on LLM-driven autoformalization. Rather than merely

asking for natural language proofs, the prompting strategy solicited formal specifications and rigorous verifications of potential counterexamples directly in the Lean theorem prover. The interaction followed a "human-guided, machine-synthesized" paradigm: prompts were engineered to request full formal proofs and standard research-style counterexample constructions. Remarkably, the LLM synthesized the vast majority of the executable Lean code, including the intricate logic required to verify sum and difference properties, while the human authors focused on high-level validation of the formalized output.

3.5 LLM-Guided First-Order Taylor Expansion for Biconjugate Operators

(Salim, 2025) leverage LLMs to tackle a technical challenge in convex analysis: deriving the sensitivity of the biconjugation operator (Fenchel and Blackett, 1953) with respect to function perturbations. The core mathematical objective is to establish the first-order Taylor expansion for the gradient of the biconjugate $(\phi + th)^{**}$. Formally, for a strictly convex function $\phi : \mathbb{R}^d \to \mathbb{R}$ and a test function h, the study aims to prove:

$$\nabla(\phi + th)^{**}(x) = \nabla\phi(x) + t\nabla h(x) + o(t) \quad (1)$$

for almost every $x \in \mathbb{R}^d$.

The research methodology employed GPT-5-pro as a symbolic reasoning assistant. The authors issued targeted queries prompting the model to generate proofs for Equation (1) and to explicate intermediate logical transitions. While the model successfully hypothesized critical lemmas and outlined the proof architecture, it exhibited "conceptual slippage" on subtle theoretical constraints, particularly regarding the implications of strong convexity. This necessitated a rigorous "human-in-the-loop" rectification process: the authors acted as discriminators, preserving valid symbolic derivations while manually correcting theoretical errors to finalize the rigorous argument.

3.6 Explicit Convergence Rates in Malliavin–Stein Analysis via LLM Guidance

(Diez et al., 2025) explores the potential of LLMs to catalyze the transition from qualitative limit theorems to quantitative approximations within the Malliavin–Stein framework (Nourdin and Peccati, 2009). Specifically, building upon the qualitative foundations laid in (Basse-O'Connor et al., 2025), the study focuses on deriving explicit convergence rates for normalized sums of Wiener-Itô integrals. Consider the variable

$$Z = X + Y,$$

where $X = I_p(f)$ and $Y = I_q(g)$ represent integrals of orders p and q, standardized such that $\mathbb{E}[Z^2] = 1$.

The primary objective is to bound the total variation distance to the standard normal distribution strictly in terms of the fourth cumulant:

$$d_{\mathrm{TV}}(Z, \mathcal{N}(0,1)) \leq \sqrt{6, \kappa_4(Z)},$$

where $\kappa_4(Z) = \mathbb{E}[Z^4] - 3$.

The prompting strategy was designed to elicit "quantitative upgrades" of existing theorems. Rather than generic queries, the authors tasked the model with rigorous derivation challenges, such as converting the qualitative Fourth Moment Theorem into explicit error bounds. The LLM-generated proofs—including the handling of complex mixed moments like $\mathbb{E}[X^3Y]$—were subjected to granular, line-by-line human verification. Furthermore, the study demonstrated the model's capability to transfer Gaussian arguments to the Poisson domain, successfully adapting the proofs to accommodate discrete jump processes and verifying the necessity of additional correction terms.

3.7 Synergizing LLMs and Evolutionary Algorithms for Mathematical Construction

In (Georgiev et al., 2025), the authors introduce AlphaEvolve, an agentic framework that synergizes Large Language Models with evolutionary algorithms to automate scientific discovery. The system functions as an iterative coding agent, specifically engineered to navigate combinatorial search spaces in search of mathematical objects with extremal quantitative properties—ranging from geometric configurations to number-theoretic constants.

Methodologically, AlphaEvolve reformulates mathematical construction as a program synthesis task. Provided with only a minimalist problem description and devoid of heuristic hints (a "zero-knowledge" initialization), the agent generates and mutates code to implement search heuristics. This process is governed by a human-in-the-loop protocol: while the agent autonomously explores the

Formalization Prompt

```
You are a mathematical formalization expert. Your task is to convert the theorem
    or lemma shown in the image into formal LaTeX code.

Requirements:
1. Output ONLY LaTeX code in a code block
2. Do NOT include any explanations, text, or markdown outside the code block
3. Do NOT use \section{} or \subsection{}
4. Do NOT use itemize, enumerate, or similar list environments
5. Write the theorem/lemma using \begin{theorem} or \begin{lemma} environment
6. Use proper mathematical notation and formatting
7. Be precise and rigorous in the formalization

Output the formalized theorem/lemma in LaTeX format.
```

Figure 1: The strict system prompt used for Stage 2 (Statement Formalization).

solution space via code execution to optimize numerical bounds, human experts are responsible for validating the numerical stability of the results and formalizing the underlying theoretical arguments.

4 Methodology and Workflow

Our benchmark framework is organized into six distinct stages, designed to clearly separate problem formalization from reasoning and to enforce human quality control at critical junctures. This workflow ensures reproducibility and prevents the propagation of errors common in automated mathematical reasoning.

The process consists of: (1) **Data Curation**, including problem extraction, context merging, and quality filtering; (2) **Automated Formalization**, using vision-capable models to convert images to standard LaTeX; (3) **Statement Verification**, ensuring the absence of hallucinations or semantic deviations; (4) **Automated Proof Generation**, utilizing frontier models with structured workload assignments; (5) **Automated Verification**, where an AI verifier checks for logical completeness; and (6) **Meta-Verification**, a human review of sampled outputs to assess system reliability.

4.1 Stage 1: Data Curation (Human)

We extracted problems from the standard textbook *Communication Complexity* (Kushilevitz and Nisan, 1997). To ensure consistency, input data was handled according to four categories:

- **Direct Screenshots:** Used for exercises clearly presented on a single page.
- **Multi-page Merging:** Applied to problems spanning multiple pages, concatenating images to maintain context.
- **Dependency Resolution:** Referenced lemmas were manually added to make each problem self-contained.
- **Manual Restoration:** Low-quality images were manually retyped into LaTeX to provide high-quality inputs for the vision models.

4.2 Stage 2: Automated Formalization (AI)

We deploy the multimodal capabilities of Claude 3.5 Sonnet to transcribe visual problem instances into structured LaTeX (see Figure 1). To minimize generation variance, we engineered a rigorous system prompt (temperature $T = 0.2$) that mandates the use of standard mathematical environments while strictly suppressing conversational artifacts. Due to space limitations, the detailed prompt specifications and the formalization procedure (Algorithm 1) are deferred to B.

4.3 Stage 3: Statement Verification (Human)

A rigorous human-in-the-loop review was conducted to validate all formalized outputs. This process prioritized the correction of specific model failure modes, including variable hallucinations

Proof Generation Prompt

```
You are a rigorous mathematical proof writer. Given the following formalized
    theorem or lemma in LaTeX, provide a complete and formal mathematical proof.

FORMALIZED STATEMENT:
{formalized_statement}

Requirements:
1. Output ONLY LaTeX code in a code block
2. Do NOT include any explanations, text, or markdown outside the code block
3. Do NOT use \section{} or \subsection{}
4. Do NOT use itemize, enumerate, or similar list environments
5. Write all equations inline using $...$ or in display mode using \[...\] or $$
    ...$$
6. Begin with \begin{proof} and end with \end{proof}
7. The proof must be rigorous, complete, and written in formal mathematical
    style
8. Include all necessary steps and justifications
9. Use proper mathematical notation and logical structure

Provide the complete formal proof in LaTeX format.
```

Figure 2: The strict system prompt used for Stage 4 (Proof Generation).

and formulaic syntax errors, while ensuring semantic equivalence between the rendered LaTeX and the original source image. Only verified, high-precision statements were propagated to the reasoning stage.

4.4 Stage 4: Automated Proof Generation (AI)

We benchmarked a suite of frontier reasoning models, including Claude-Sonnet-4.5, GPT-5-Thinking, Gemini-3-Pro, and Grok-4. To accommodate the heterogeneous latency profiles and computational costs of these architectures, we engineered a rigorous execution framework comprising a partitioned workload assignment and a dynamic time-allocation protocol.

Stratified Workload Execution. To minimize experimental variance, the evaluation workload was stratified by textbook chapter. Each human researcher was assigned a specific mathematical domain and a fixed model architecture. This isolation ensures a consistent evaluation environment for all problems within a topic, thereby reducing confounding variables arising from inconsistent human-model interactions.

Adaptive Double-Timing Strategy. Static timeout thresholds are notoriously inefficient for mathematical reasoning: they typically under-serve complex deduction chains while over-allocating resources to trivial tasks. We implemented an adaptive resource escalation strategy (Algorithm 4). The process initiates with an aggressive baseline ($T_{\text{init}} = 300\text{s}$).

Crucially, a non-response is not immediately classified as a failure; instead, the system triggers a back-off mechanism that doubles the computational window ($T_{\text{new}} \leftarrow T_{\text{old}} \times 2$) and retries the generation. This escalation iterates until a valid proof is synthesized or the hard ceiling ($T_{\max} = 1200\text{s}$) is reached, effectively balancing system throughput with the need for deep reasoning time. The generation process is governed by a strict system prompt (see Figure 2) to enforce formal rigor.

Qualitative Divergence in Proof Styles. Beyond quantitative runtime metrics, we observed distinct stylistic archetypes across the evaluated models, which significantly impact human verification load. **Narrative-Driven:** GPT-5-Thinking and Grok-4

Verification Prompt

```
You are a mathematical proof verification expert. Your task is to verify whether
     the provided proof is correct and complete.

FORMALIZED STATEMENT:
{formalized_statement}

PROPOSED PROOF:
{proof}

Carefully analyze the proof and determine if it is:
1. Mathematically correct (no logical errors)
2. Complete (all steps are justified)
3. Rigorous (meets formal mathematical standards)

You must respond with ONLY a single digit:
- Output "1" if the proof is correct, complete, and rigorous
- Output "0" if the proof has any errors, gaps, or issues

Do not provide any explanation. Output only "0" or "1".
```

Figure 3: The strict system prompt used for Stage 5 (Automated Verification of Proof Correctness).

exhibit a preference for high-level semantic reasoning, constructing proofs primarily in natural English with sparse notation. **Symbol-Centric:** Claude-Sonnet-4.5 prioritizes dense symbolic manipulation and explicit step-by-step derivations. **Formal-Structured:** Conversely, Gemini-3-Pro adopts a rigorous academic formalism, generating outputs that closely mimic the structured, lemma-heavy style of graduate-level textbooks.

4.5 Stage 5: Automated Verification (AI)

Given the extensive scale of the benchmark, we deployed Claude-3.5-Sonnet as a scalable verification engine. The model acts as an adjudicator, evaluating each generated proof against its corresponding formalized statement to yield a binary validity signal. The verification logic audits the proof for correctness, completeness, and rigor, as detailed in Algorithm 2. The specific prompt structure used for this adjudication is provided in Figure 3.

4.6 Stage 6: Meta-Verification (Human)

To validate the reliability of the automated verifier, we implemented a human-in-the-loop audit protocol. We performed a random sampling of 20% of all generated triplets (Problem, Proof, Verdict) for expert review. Let Δ denote the alignment discrepancy between the model-based verdicts and human ground truth on this subset. We established a strict tolerance threshold: if $\Delta \leq 20\%$, the automated assessment is deemed calibrated, and the results for the full dataset are ratified. Conversely, if the divergence exceeds this bound, the verification batch is rejected, triggering a re-evaluation cycle.

5 Conclusion

We present a comprehensive benchmark of frontier LLMs on graduate-level mathematical reasoning, with a specific focus on Communication Complexity. By decoupling formalization from deduction, our multi-stage pipeline proved essential for isolating distinct failure modes. The results reveal a stark disparity in reliability: while select models consistently deliver rigorous formal arguments, others exhibit instability marked by logical incoherence and hallucinations. Ultimately, while current models demonstrate promising competence in proof synthesis, robustness remains a critical bottleneck. We position this evaluation as a foundation for future research aimed at enhancing the reliability of automated mathematical reasoning.

Limitations

During the evaluation, we observe that certain models experience connection timeouts and fail to generate responses within the designated time window. These instances are categorized as failures in our statistics.

References

Boris Alexeev and Dustin G Mixon. 2025. Forbidden sidon subsets of perfect difference sets, featuring a human-assisted proof. *arXiv preprint arXiv:2510.19804*.

Andreas Basse-O'Connor, David Kramer-Bang, and Clement Svendsen. 2025. Fourth-moment theorems for sums of multiple integrals. *arXiv preprint arXiv:2502.03596*.

Emanuele Bugliarello, H. Hernan Moraldo, Ruben Villegas, Mohammad Babaeizadeh, Mohammad Taghi Saffar, Han Zhang, Dumitru Erhan, Vittorio Ferrari, Pieter-Jan Kindermans, and Paul Voigtlaender. 2023. Storybench: A multifaceted benchmark for continuous story visualization. In *Advances in Neural Information Processing Systems*, volume 36, pages 78095–78125. Curran Associates, Inc.

François Charton, Jordan S Ellenberg, Adam Zsolt Wagner, and Geordie Williamson. 2024. Patternboost: Constructions in mathematics with a little help from ai. *arXiv preprint arXiv:2411.00566*.

Yubin Chen, Xuyang Guo, Zhenmei Shi, Zhao Song, and Jiahao Zhang. 2025. T2vworldbench: A benchmark for evaluating world knowledge in text-to-video generation. *arXiv preprint arXiv:2507.18107*.

Jaemin Cho, Abhay Zala, and Mohit Bansal. 2023. Dall-eval: Probing the reasoning skills and social biases of text-to-image generation models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3043–3054.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, and 1 others. 2021. Advancing mathematics by guiding human intuition with ai. *Nature*.

Charles-Philippe Diez, Luis da Maia, and Ivan Nourdin. 2025. Mathematical research with gpt-5: a malliavin-stein experiment. *arXiv preprint arXiv:2509.03065*.

Michael Douglas, Subramanian Lakshminarasimhan, and Yidi Qi. 2022. Numerical calabi-yau metrics from holomorphic networks. In *Mathematical and Scientific Machine Learning*.

Paul Erdős. 1975. Problems and results in graph theory and combinatorial analysis. *Proc. British Combinatorial Conj., 5th*, pages 169–192.

Paul Erdös. 1977. Problems and results on combinatorial number theory iii. In *Number Theory Day*, pages 43–72, Berlin, Heidelberg. Springer Berlin Heidelberg.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, and 1 others. 2022. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*.

Moran Feldman and Amin Karbasi. 2025. G\" odel test: Can large language models solve easy conjectures? *arXiv preprint arXiv:2509.18383*.

Werner Fenchel and Donald W Blackett. 1953. *Convex cones, sets, and functions*. Princeton University, Department of Mathematics, Logistics Research Project.

Weixi Feng, Jiachen Li, Michael Saxon, Tsu-jui Fu, Wenhu Chen, and William Yang Wang. 2024. Tc-bench: Benchmarking temporal compositionality in text-to-video and image-to-video generation. *arXiv preprint arXiv:2406.08656*.

Xingyu Fu, Muyu He, Yujie Lu, William Yang Wang, and Dan Roth. 2024. Commonsense-t2i challenge: Can text-to-image generation models understand commonsense? In *First Conference on Language Modeling*.

Bogdan Georgiev, Javier Gómez-Serrano, Terence Tao, and Adam Zsolt Wagner. 2025. Mathematical exploration and discovery at scale. *arXiv preprint arXiv:2511.02864*.

Xuyang Guo, Jiayan Huo, Zhenmei Shi, Zhao Song, Jiahao Zhang, and Jiale Zhao. 2025. T2vphysbench: A first-principles benchmark for physical consistency in text-to-video generation. *arXiv preprint arXiv:2505.00337*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. 2021. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7514–7528.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. 2022. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646.

Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A Smith. 2023. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20406–20417.

Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. 2023. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. *Advances in Neural Information Processing Systems*, 36:78723–78747.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, and 1 others. 2024a. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21807–21818.

Ziqi Huang, Fan Zhang, Xiaojie Xu, Yinan He, Jiashuo Yu, Ziyue Dong, Qianli Ma, Nattapol Chanpaisit, Chenyang Si, Yuming Jiang, and 1 others. 2024b. Vbench++: Comprehensive and versatile benchmark suite for video generative models. *arXiv preprint arXiv:2411.13503*.

Paata Ivanisvili and Xinyuan Xie. 2025. Counterexample to majority optimality in nicd with erasures. *arXiv preprint arXiv:2510.20013*.

Uijeong Jang and Ernest K Ryu. 2025. Point convergence of nesterov's accelerated gradient method: An ai-assisted proof. *arXiv preprint arXiv:2510.23513*.

Pengliang Ji, Chuyang Xiao, Huilin Tai, and Mingxiao Huo. 2024. T2vbench: Benchmarking temporal dynamics for text-to-video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 5325–5335.

Xue Jiang, Yihong Dong, Lecheng Wang, Zheng Fang, Qiwei Shang, Ge Li, Zhi Jin, and Wenpin Jiao. 2024. Self-planning code generation with large language models. *ACM Transactions on Software Engineering and Methodology*.

Eyal Kushilevitz and Noam Nisan. 1997. *Communication Complexity*. Cambridge University Press, Cambridge, UK.

Chao Lei, Yanchuan Chang, Nir Lipovetzky, and Krista A Ehinger. 2025. Planning-driven programming: A large language model programming workflow. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12647–12684.

Baiqi Li, Zhiqiu Lin, Deepak Pathak, Jiayao Emily Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. 2024a. Genai-bench: A holistic benchmark for compositional text-to-visual generation. In *Synthetic Data for Computer Vision Workshop@ CVPR 2024*.

Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. 2024b. Seedbench: Benchmarking multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Mingxiang Liao, Hannan Lu, Xinyu Zhang, Fang Wan, Tianyu Wang, Yuzhong Zhao, Wangmeng Zuo, Qixiang Ye, and Jingdong Wang. 2024. Evaluation of text-to-video generation models: A dynamics perspective. In *Advances in Neural Information Processing Systems*, volume 37, pages 109790–109816. Curran Associates, Inc.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167.

Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. 2024a. Mathbench: Evaluating the theory and application proficiency of llms with a hierarchical mathematics benchmark. *arXiv preprint arXiv:2405.12209*.

Yaofang Liu, Xiaodong Cun, Xuebo Liu, Xintao Wang, Yong Zhang, Haoxin Chen, Yang Liu, Tieyong Zeng, Raymond Chan, and Ying Shan. 2024b. Evalcrafter: Benchmarking and evaluating large video generation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22139–22149.

Yuanxin Liu, Lei Li, Shuhuai Ren, Rundong Gao, Shicheng Li, Sishuo Chen, Xu Sun, and Lu Hou. 2023. Fetv: A benchmark for fine-grained evaluation of open-domain text-to-video generation. In *Advances in Neural Information Processing Systems*, volume 36, pages 62352–62387. Curran Associates, Inc.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2023. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*.

Fanqing Meng, Wenqi Shao, Lixin Luo, Yahong Wang, Yiran Chen, Quanfeng Lu, Yue Yang, Tianshuo Yang, Kaipeng Zhang, Yu Qiao, and 1 others. 2024. Phybench: A physical commonsense benchmark for evaluating text-to-image models. *arXiv preprint arXiv:2406.11802*.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294.

Yurii Nesterov. 1983. A method for solving the convex programming problem with convergence rate o (1/k2). In *Dokl akad nauk Sssr*, volume 269, page 543.

Ivan Nourdin and Giovanni Peccati. 2009. Stein's method on wiener chaos. *Probability Theory and Related Fields*, 145(1):75–118.

Vitali Petsiuk, Alexander E Siemenn, Saisamrit Surbehera, Zad Chin, Keith Tyser, Gregory Hunter, Arvind Raghavan, Yann Hicke, Bryan A Plummer, Ori Kerret, and 1 others. 2022. Human evaluation of text-to-image models on a multi-task benchmark. *arXiv preprint arXiv:2211.12112*.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, and 1 others. 2024. Mathematical discoveries from program search with large language models. *Nature*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, and 1 others. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494.

Adil Salim. 2025. Accelerating mathematical research with language models: A case study of an interaction with gpt-5-pro on a convex analysis problem. *arXiv preprint arXiv:2510.26647*.

Simon Sidon. 1932. Ein satz über trigonometrische polynome und seine anwendung in der theorie der fourier-reihen. *Mathematische Annalen*, 106(1):536–539.

Kaiyue Sun, Kaiyi Huang, Xian Liu, Yue Wu, Zihan Xu, Zhenguo Li, and Xihui Liu. 2024. T2v-compbench: A comprehensive benchmark for compositional text-to-video generation. *arXiv preprint arXiv:2407.14505*.

Thomas Unterthiner, Sjoerd Van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. 2018. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*.

Adam Zsolt Wagner. 2021. Constructions in combinatorics via neural networks. *arXiv preprint arXiv:2104.14516*.

Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. 2021. Naturalproofs: Mathematical theorem proving in natural language. *arXiv preprint arXiv:2104.01112*.

Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Stan Weixian Lei, Yuchao Gu, Yufei Shi, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. 2023. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7623–7633.

Xindi Wu, Dingli Yu, Yangsibo Huang, Olga Russakovsky, and Sanjeev Arora. 2024. Conceptmix: A compositional image generation benchmark with controllable difficulty. In *NeurIPS 2024 Workshop on Compositional Learning: Perspectives, Methods, and Paths Forward*.

Ke Yang. 2007. On the (im) possibility of non-interactive correlation distillation. *Theoretical Computer Science*, 382(2):157–166.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, and 1 others. 2024. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. 2023. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490*.

Yaqi Zhao, Xiaochen Wang, Li Dong, Wentao Zhang, and Yuhui Yuan. 2025. Demystifying numerosity in diffusion models–limitations and remedies. *arXiv preprint arXiv:2510.11117*.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *ICLR*.

Appendix

Roadmap. Section A provides potential risks. In Section B, we present the algorithm of the whole procedure. From Section C to Section O, we show our detailed problem statement and proofs generated by LLMs.

A Potential Risks

The primary societal risk associated with our work lies in the potential misuse of automated reasoning systems in educational settings. As shown in our benchmarks, frontier LLMs are achieving proficiency in graduate-level mathematical problems. While this signifies progress in AI, it simultaneously poses a challenge to traditional assessment methods in academia. Students might employ such systems to complete coursework or take-home examinations, potentially undermining the integrity of academic evaluation. Educators may need to adapt by shifting towards in-person assessments or designing problems that require novel reasoning patterns unseen in training data.

B More Algorithm

Algorithm 1 Formalization Procedure

data structure BENCHMARKSYSTEM
procedure FORMALIZE
    $\mathcal{S} \leftarrow \emptyset$
    for each image $I_j \in \mathcal{I}$ **do**
        response $\leftarrow M_{\text{vision}}.\text{GENERATE}(I_j, P_{\text{form}}, T)$
        $S_j \leftarrow \text{EXTRACTLATEX}(\text{response})$
        $\mathcal{S} \leftarrow \mathcal{S} \cup \{S_j\}$
    end for
end procedure
end data structure

Algorithm 2 Proof Verification Procedure

data structure BENCHMARKSYSTEM
procedure VERIFYPROOFS
    $\mathcal{V} \leftarrow \emptyset$
    for each $(S_j, M_i) \in \text{keys}(\mathcal{R})$ **do**
        $R_{ij} \leftarrow \mathcal{R}[S_j, M_i]$
        prompt $\leftarrow P_{\text{verify}}.\text{FORMAT}(S_j, R_{ij})$
        response $\leftarrow M_{\text{verifier}}.\text{GENERATE}(\text{prompt}, T)$
        $v_{ij} \leftarrow \text{EXTRACTBINARYRESULT}(\text{response})$ ▷ Parse 0 or 1
        $\mathcal{V}[S_j, M_i] \leftarrow v_{ij}$
    end for
    $\mathcal{V}_{\text{sample}} \leftarrow \text{RANDOMSAMPLE}(\mathcal{V}, k = 0.2|\mathcal{V}|)$
    $\mathcal{V}_{\text{human}} \leftarrow \text{HUMANVERIFY}(\mathcal{V}_{\text{sample}})$
    $\text{VALIDATEAGREEMENT}(\mathcal{V}_{\text{sample}}, \mathcal{V}_{\text{human}})$
end procedure
end data structure

Algorithm 3 Benchmark System Data Structure

data structure BENCHMARKSYSTEM
members
  $\mathcal{I} = \{I_1, \ldots, I_n\}$ ▷ Input images of problems
  $\mathcal{S} = \{S_1, \ldots, S_n\}$ ▷ Formalized LaTeX statements
  $\mathcal{M} = \{M_1, \ldots, M_k\}$ ▷ Set of models to evaluate
  $\mathcal{R}$ ▷ Generated proofs, indexed by (S_j, M_i)
  $\mathcal{F}$ ▷ Failed proof attempts, indexed by (S_j, M_i)
  $\mathcal{V}$ ▷ Verification results $\{0, 1\}$, indexed by (S_j, M_i)
  T_0 ▷ Initial timeout (e.g., 300s)
  K ▷ Maximum timeout multiplier (e.g., 4)
  T ▷ Generation temperature (e.g., 0.2)
  $P_{\text{form}}, P_{\text{proof}}, P_{\text{verify}}$ ▷ Prompt templates
  $M_{\text{vision}}, M_{\text{verifier}}$ ▷ Models for formalization and verification
end members

Algorithm 4 Proof Generation Procedure with Doubling Timeout

data structure BENCHMARKSYSTEM
procedure GENERATEPROOFS
  $\mathcal{R} \leftarrow \emptyset$, $\mathcal{F} \leftarrow \emptyset$
  for each model $M_i \in \mathcal{M}$ **do**
    $\mathcal{S}_{\text{pending}} \leftarrow \mathcal{S}$ ▷ All problems for this model
    $t \leftarrow T_0$ ▷ Start with base timeout
    while $t \leq K \cdot T_0$ & $\mathcal{S}_{\text{pending}} \neq \emptyset$ **do**
      $\mathcal{S}_{\text{retry}} \leftarrow \emptyset$ ▷ Problems that timeout in this round
      for each $S_j \in \mathcal{S}_{\text{pending}}$ **do**
        $\text{prompt} \leftarrow P_{\text{proof}}.\text{FORMAT}(S_j)$
        $(R_{ij}, \sigma) \leftarrow M_i.\text{GENERATE}(\text{prompt}, T, \text{timeout} = t)$
        if $\sigma = \text{SUCCESS}$ **then**
          $\mathcal{R}[S_j, M_i] \leftarrow R_{ij}$
        else if $\sigma = \text{TIMEOUT}$ **then**
          $\mathcal{S}_{\text{retry}} \leftarrow \mathcal{S}_{\text{retry}} \cup \{S_j\}$
        else
          $\mathcal{F}[S_j, M_i] \leftarrow \sigma$
        end if
      end for
      $\mathcal{S}_{\text{pending}} \leftarrow \mathcal{S}_{\text{retry}}$
      $t \leftarrow 2t$ ▷ Double timeout
    end while
    for each $S_j \in \mathcal{S}_{\text{pending}}$ **do**
      $\mathcal{F}[S_j, M_i] \leftarrow \text{MAXTIMEOUTEXCEEDED}$
    end for
  end for
end procedure
end data structure

C Problems in Chapter 1

C.1 Problem 1

C.1.1 Problem Statement

Theorem C.1 (Exercise 1.5)**.** *Alice and Bob hold subsets $x, y \subseteq \{1, \ldots, n\}$, respectively, and they wish to compute AVG(x, y), which is defined as the average number in the multiset $x \cup y$. Then $D(\text{AVG}) = O(\log n)$. (Note that the average need not be an integer.)*

C.1.2 GPT

Proof. Let Alice hold $x \subseteq \{1, \ldots, n\}$ and Bob hold $y \subseteq \{1, \ldots, n\}$. Define

$$S_x = \sum_{i \in x} i, \quad S_y = \sum_{i \in y} i, \quad m_x = |x|, \quad m_y = |y|.$$

Then the multiset union $x \cup y$ contains each element of x and each element of y with multiplicity, so its total sum is $S_x + S_y$ and its total size is $m_x + m_y$. Hence

$$\text{AVG}(x, y) = \frac{S_x + S_y}{m_x + m_y},$$

with the convention that if $m_x + m_y = 0$ then the average is undefined; in this case any fixed default output may be specified and communicated with 0 bits, so we assume $m_x + m_y \geq 1$.

Consider the following deterministic two-way protocol. Alice computes (S_x, m_x) and sends them to Bob. Bob computes (S_y, m_y), forms $S = S_x + S_y$ and $m = m_x + m_y$, and outputs S/m.

It remains to bound the number of bits communicated. Since $0 \leq m_x \leq n$, the value m_x can be encoded in $\lceil \log_2(n+1) \rceil$ bits. Also

$$0 \leq S_x = \sum_{i \in x} i \leq \sum_{i=1}^{n} i = \frac{n(n+1)}{2},$$

so S_x can be encoded in $\left\lceil \log_2\left(\frac{n(n+1)}{2} + 1\right) \right\rceil$ bits. Therefore the total communication from Alice to Bob is at most

$$\lceil \log_2(n+1) \rceil + \left\lceil \log_2\left(\frac{n(n+1)}{2} + 1\right) \right\rceil = O(\log n).$$

The protocol is correct because Bob outputs $(S_x + S_y)/(m_x + m_y) = \text{AVG}(x, y)$, and it is deterministic with $O(\log n)$ communication. Hence $D(\text{AVG}) = O(\log n)$. □

C.1.3 Claude

Proof. We prove that the deterministic communication complexity of computing AVG(x, y) is $O(\log n)$.

First, observe that for subsets $x, y \subseteq \{1, \ldots, n\}$, the average of the multiset $x \cup y$ is given by

$$\text{AVG}(x, y) = \frac{\sum_{i \in x} i + \sum_{j \in y} j}{|x| + |y|}.$$

We establish the following protocol:

Step 1: Alice computes $S_x = \sum_{i \in x} i$ and $|x|$. Since $S_x \leq n \cdot n = n^2$, Alice can represent S_x using $O(\log n)$ bits. Similarly, $|x| \leq n$ requires $O(\log n)$ bits. Alice sends both values to Bob, using a total of $O(\log n)$ bits.

Step 2: Bob receives S_x and $|x|$ from Alice. Bob computes $S_y = \sum_{j \in y} j$ and $|y|$ locally.

Step 3: Bob computes the final result:

$$\text{AVG}(x, y) = \frac{S_x + S_y}{|x| + |y|}.$$

Note that $|x| + |y| > 0$ whenever at least one of the sets is nonempty. If both sets are empty, we adopt the convention that $\text{AVG}(\emptyset, \emptyset) = 0$ or is undefined, which can be handled separately with no communication.

Now we verify the communication cost. Alice sends:

- S_x, which requires at most $\lceil \log_2(n^2 + 1) \rceil = O(\log n)$ bits.
- $|x|$, which requires at most $\lceil \log_2(n + 1) \rceil = O(\log n)$ bits.

Therefore, the total communication is $O(\log n) + O(\log n) = O(\log n)$ bits.

Since Bob can compute the exact value of $\text{AVG}(x, y)$ after receiving this information, and the protocol uses only $O(\log n)$ bits of communication, we conclude that $D(\text{AVG}) = O(\log n)$. □

C.1.4 Gemini

Proof. Let $x, y \subseteq \{1, \ldots, n\}$ be the inputs held by Alice and Bob, respectively. The goal is to compute the average of the numbers in the multiset $x \cup y$. Let $S_x = \sum_{a \in x} a$ denote the sum of the elements in Alice's set, and let $N_x = |x|$ denote the cardinality of Alice's set. Similarly, let $S_y = \sum_{b \in y} b$ and $N_y = |y|$ be the sum and cardinality of Bob's set. The average of the multiset $x \cup y$ is given by the formula:

$$\text{AVG}(x, y) = \frac{S_x + S_y}{N_x + N_y}.$$

We define a deterministic protocol to compute this value. In the first step, Alice computes S_x and N_x locally. We analyze the number of bits required to represent these values. Since $x \subseteq \{1, \ldots, n\}$, the cardinality satisfies $0 \leq N_x \leq n$. Therefore, N_x can be encoded using $B_N = \lceil \log_2(n + 1) \rceil$ bits. The sum S_x is bounded by the sum of all elements in the universe $\{1, \ldots, n\}$, so we have:

$$0 \leq S_x \leq \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Consequently, S_x can be encoded using $B_S = \lceil \log_2 \left(\frac{n(n+1)}{2} + 1 \right) \rceil$ bits.

Alice sends the binary representations of N_x and S_x to Bob. The total communication cost for this transmission is $B_N + B_S$. Observing the asymptotic behavior, we have $B_N = O(\log n)$ and $B_S = O(\log(n^2)) = O(2 \log n) = O(\log n)$. Thus, the total number of bits sent by Alice is $O(\log n)$.

Upon receiving N_x and S_x, Bob computes N_y and S_y from his private input y. Bob then computes the total sum $S_{\text{total}} = S_x + S_y$ and the total count $N_{\text{total}} = N_x + N_y$. Assuming the multiset is non-empty (i.e., $N_{\text{total}} > 0$), Bob computes the value $\mu = S_{\text{total}} / N_{\text{total}}$. This value μ is the exact output of $\text{AVG}(x, y)$.

If the definition of communication complexity for this problem requires the output to be known by both parties, Bob must send μ back to Alice. The value μ is a rational number p/q where $p = S_{\text{total}} \leq n(n+1)$ and $q = N_{\text{total}} \leq 2n$. The binary representation of this rational number (or the pair (p, q) in lowest terms) requires $O(\log(n^2)) + O(\log(2n)) = O(\log n)$ bits.

In either case, the total number of bits communicated is the sum of the bits for N_x, S_x and potentially the result μ, all of which are $O(\log n)$. Therefore, the deterministic communication complexity is $D(\text{AVG}) = O(\log n)$. □

C.1.5 Grok

Proof. The multiset

$$x \cup y$$

is formed by adding the multiplicities: each element in

$$x \cap y$$

has multiplicity 2, each element in

$$x \setminus y$$

or

$$y \setminus x$$

has multiplicity 1. Thus, the size of the multiset is

$$k = |x| + |y|$$

and the sum of its elements is

$$\sigma = \sum_{i \in x} i + \sum_{i \in y} i$$

. The average is

$$\mathrm{AVG}(x, y) = \sigma / k$$

.

Alice locally computes

$$s = |x| \leq n$$

and

$$\sigma_x = \sum_{i \in x} i \leq n(n+1)/2 < 2^{2 \log_2 n + 1}$$

.

Bob locally computes

$$t = |y| \leq n$$

and

$$\sigma_y = \sum_{i \in y} i \leq n(n+1)/2 < 2^{2 \log_2 n + 1}$$

.

Alice sends the values

$$s$$

and

$$\sigma_x$$

to Bob. Sending

$$s$$

requires

$$O(\log n)$$

bits, and sending

$$\sigma_x$$

requires

$$O(\log n)$$

bits.

Bob computes

$$k = s + t$$

and

$$\sigma = \sigma_x + \sigma_y$$

, then computes

$$\mathrm{AVG}(x, y) = \sigma / k$$

and outputs it.

The protocol uses

$$O(\log n)$$

bits of communication and correctly computes the average (which need not be an integer).

Thus,

$$D(\mathrm{AVG}) = O(\log n)$$

. □

C.2 Problem 2

C.2.1 Problem Statement

Theorem C.2. *Give an $O(\log n)$ bit protocol for* MED.

C.2.2 GPT

Proof. We are tasked with providing an $O(\log n)$ bit protocol for the MED problem, which involves determining the median of two sorted arrays $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$ of size n each.

Let the goal be to find the median element of the combined array $C = (a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n)$, which is the element at position $\lceil \frac{2n}{2} \rceil = n$ in the sorted order of C. The algorithm must efficiently find the median using an $O(\log n)$ bit protocol, i.e., by exchanging a logarithmic number of bits between two parties.

We first observe that a brute-force approach that merges the two arrays and selects the nth element requires $O(n)$ time and $O(n)$ bits. However, a more efficient approach using a binary search strategy will lead to the desired logarithmic complexity.

We begin by performing a binary search on one of the arrays, say A, to locate the correct partition between the two arrays. The binary search is conducted over the range of indices of A, i.e., the index set $\{1, 2, \ldots, n\}$. Let i be the index in A at which the partition occurs, and let $j = n - i$ be the corresponding index in B. The objective is to select i such that the elements in A and B are divided into two parts where the elements in the left part are smaller than those in the right part.

For the partition to be correct, the following conditions must hold: 1. $a_i \geq b_{j+1}$, ensuring that the largest element in the left partition of A is greater than or equal to the smallest element in the right partition of B. 2. $b_j \geq a_{i+1}$, ensuring that the largest element in the left partition of B is greater than or equal to the smallest element in the right partition of A.

If both conditions are satisfied, then the median is given by $\max(a_i, b_j)$.

If the conditions are not satisfied, we adjust the binary search based on the comparison of a_i and b_j. If $a_i < b_j$, it indicates that we need to increase i, and thus move the partition towards the right. If $b_j < a_i$, we decrease i, moving the partition towards the left.

The binary search halts when a valid partition is found. Since the number of steps in the binary search is $O(\log n)$, the total number of bits exchanged for each step is also $O(\log n)$.

Thus, the protocol terminates after $O(\log n)$ steps, with each step requiring $O(\log n)$ bits of communication. Therefore, the total bit complexity of the protocol is $O(\log n)$.

□

C.2.3 Claude

Proof. We present a communication protocol for the MED problem that uses $O(\log n)$ bits, where Alice and Bob each hold a sorted array of n elements and wish to determine the median of the combined $2n$ elements.

The protocol proceeds by binary search on the possible median values. Let $A[1], \ldots, A[n]$ denote Alice's sorted array and $B[1], \ldots, B[n]$ denote Bob's sorted array. The median of the combined array is the n-th smallest element.

Alice and Bob maintain an interval $[L, R]$ representing the possible range of the median. Initially, $L = \min(A[1], B[1])$ and $R = \max(A[n], B[n])$. In each round, they perform the following steps:

Alice computes $m = \frac{L+R}{2}$ and determines $i_A = |\{j : A[j] \leq m\}|$, the number of elements in her array at most m. She sends i_A to Bob using $\lceil \log(n+1) \rceil$ bits.

Bob computes $i_B = |\{j : B[j] \leq m\}|$, the number of elements in his array at most m. He calculates $i_A + i_B$, the total number of elements at most m in the combined array.

If $i_A + i_B < n$, then the median is greater than m, so Bob sends 0 to Alice (one bit) and they update $L = m$.

If $i_A + i_B \geq n$, then the median is at most m, so Bob sends 1 to Alice (one bit) and they update $R = m$.

The interval $[L, R]$ is halved in each round. After $O(\log U)$ rounds, where U is the universe size, the interval contains a single value, which is the median. However, we can improve this by working directly with array indices rather than values.

More precisely, Alice and Bob perform binary search on the index of the median in Alice's array. Let $k \in \{0, 1, \ldots, n\}$ denote a candidate where the median could be between $A[k]$ and $A[k+1]$ (with $A[0] = -\infty$ and $A[n+1] = +\infty$). For each candidate k, Bob determines the corresponding index ℓ in his array such that $A[k] \leq B[\ell] < A[k+1]$. The median is the n-th smallest element, so we need $k + \ell = n$.

Alice sends a candidate index k using $\lceil \log(n+1) \rceil$ bits. Bob computes the number of elements in his array less than $A[k]$, denoted ℓ, and checks if $k + \ell \geq n$. Bob sends back one bit indicating whether to search in the left or right half.

After $O(\log n)$ rounds of binary search, they identify the correct index k such that the n-th smallest element is either $A[k]$ or some element in Bob's array. A final $O(\log n)$ bit exchange determines the exact median.

The total communication is $O(\log n) \cdot (\log n + 1) = O(\log^2 n)$ bits for the naive approach, but with careful implementation using exponential search and maintaining tighter bounds, the protocol achieves $O(\log n)$ bits total communication. □

C.2.4 Gemini

Proof. Let $x, y \subseteq \{1, \ldots, n\}$ be the sets held by Alice and Bob respectively. We interpret these as sorted arrays A and B where A_i is the i-th smallest element of x and B_j is the j-th smallest element of y. Let $k = |x| + |y|$. We seek the element of rank $\lceil k/2 \rceil$ in the multiset $A \cup B$. Without loss of generality, assume $|x| = |y| = m$ and we seek the m-th element (the lower median) of $2m$ elements. We assume m and n are powers of 2 for simplicity; the argument extends to general cases with constant overhead.

We describe a deterministic protocol that maintains a search range for the indices of A and B, and a search range for the value of the median.

Protocol State: Let $I_A \subseteq [1, m]$ and $I_B \subseteq [1, m]$ be the current ranges of candidate indices in A and B. Initially $I_A = I_B = [1, m]$. Let $V = [v_{min}, v_{max}] \subseteq [1, n]$ be the current range of possible values for the median. Initially $V = [1, n]$. In each round, the protocol reduces the size of either the index ranges or the value range by half.

Protocol Steps: In each round, performed while $|I_A| > 1$ and $|V| > 1$:

1. Let mid_A and mid_B be the middle indices of the current ranges I_A and I_B.
2. Let $v_{mid} = \lfloor (v_{min} + v_{max})/2 \rfloor$ be the middle of the value range.
3. Alice computes a bit $b_A = 1$ if $A[mid_A] > v_{mid}$, and 0 otherwise.
4. Bob computes a bit $b_B = 1$ if $B[mid_B] > v_{mid}$, and 0 otherwise.
5. Alice and Bob exchange b_A and b_B (2 bits of communication).
6. They update their ranges based on the pair (b_A, b_B):
   - **Case 1:** $(0, 0)$**.** Both $A[mid_A] \leq v_{mid}$ and $B[mid_B] \leq v_{mid}$. This implies that the lower halves of the current candidate arrays in A and B (elements at indices $\leq mid_A$ in I_A and $\leq mid_B$ in I_B) are all $\leq v_{mid}$. The total count of such elements is at least $|I_A|/2 + |I_B|/2$. Since $|I_A| = |I_B|$, this count is $|I_A|$. In the current recursive subproblem finding the median of the remaining elements, the target rank is roughly half the total elements. Since we have identified a sufficient number of elements $\leq v_{mid}$, the median must be $\leq v_{mid}$. **Update:** $V \leftarrow [v_{min}, v_{mid}]$. The index ranges I_A, I_B remain unchanged.
   - **Case 2:** $(1, 1)$**.** Both $A[mid_A] > v_{mid}$ and $B[mid_B] > v_{mid}$. Symmetrically, there are sufficient elements $> v_{mid}$ to guarantee the median is $> v_{mid}$. **Update:** $V \leftarrow [v_{mid} + 1, v_{max}]$. The index ranges I_A, I_B remain unchanged.

- **Case 3:** $(0,1)$**.** $A[mid_A] \leq v_{mid} < B[mid_B]$. This implies $A[mid_A] < B[mid_B]$. By the standard property of medians of sorted arrays, the median of $A \cup B$ cannot reside in the lower half of I_A (indices $\leq mid_A$) nor in the upper half of I_B (indices $> mid_B$). We can discard these portions. **Update:** $I_A \leftarrow$ upper half of I_A, $I_B \leftarrow$ lower half of I_B. The value range V remains unchanged.
- **Case 4:** $(1,0)$**.** $A[mid_A] > v_{mid} \geq B[mid_B]$. This implies $B[mid_B] < A[mid_A]$. We discard the lower half of I_B and the upper half of I_A. **Update:** $I_A \leftarrow$ lower half of I_A, $I_B \leftarrow$ upper half of I_B. The value range V remains unchanged.

Complexity Analysis: Let S_k be the size of the index ranges and R_k be the size of the value range at step k. Initially $S_0 = m \leq n$ and $R_0 = n$. In each step, we transmit 2 bits. If Case 1 or 2 occurs, $R_{k+1} = R_k/2$. If Case 3 or 4 occurs, $S_{k+1} = S_k/2$. The protocol terminates when $S_k = 1$ or $R_k = 1$. The maximum number of times the value range can shrink is $\log n$. The maximum number of times the index range can shrink is $\log m \leq \log n$. Thus, the total number of rounds is bounded by $\log n + \log m \leq 2 \log n$. The total communication cost is $2 \cdot (2 \log n) = O(\log n)$ bits.

Correctness: The invariant is that the global median lies within the values of $A[I_A] \cup B[I_B]$ restricted to V. In Cases 1 and 2, we determine that the median lies in the lower or upper half of V based on the counts of elements known to be smaller or larger than v_{mid}. In Cases 3 and 4, we perform the standard prune-and-search reduction for the median of two sorted arrays, which preserves the median. Since the ranges monotonically decrease and cover the solution space, the protocol converges to the correct median.

Therefore, there exists a deterministic protocol for MED using $O(\log n)$ bits. □

C.2.5 Grok

Proof. We describe a deterministic protocol that computes the median of the multiset $x \cup y$ (with multiplicities added for common elements) using $O(\log n)$ bits of communication.

Alice and Bob first exchange $|x|$ and $|y|$ ($O(\log n)$ bits). Both compute $k = |x| + |y|$ and $r = \lfloor (k+1)/2 \rfloor$.

The median is the smallest m such that the number of elements $\leq m$ in the multiset is at least r.

Both initialize $L = 1$ and $R = n$.

While $L < R$:

Both compute $\text{mid} = \lfloor (L+R)/2 \rfloor$.

Bob computes $q = |\{j \in y \mid j \leq \text{mid}\}|$ and sends q to Alice ($O(\log n)$ bits).

Alice computes $p = |\{i \in x \mid i \leq \text{mid}\}|$.

If $p + q \geq r$, both set $R = \text{mid}$.

Else, both set $L = \text{mid} + 1$.

Alice sends the choice (keep left or right, $O(1)$ bits) to Bob.

There are $O(\log n)$ iterations, for total $O(\log n)$ bits.

The loop terminates with $L = R = m$, the median.

Correctness: The interval $[L, R]$ always contains the defined median, and each step preserves this invariant while halving the interval length on average.

Thus, there exists an $O(\log n)$-bit protocol for MED. □

C.3 Problem 3

C.3.1 Problem Statement

Theorem C.3 (Clique vs. Independent Set Communication Complexity)**.** *Let G be a graph on n vertices. Define the "clique vs. independent set" problem with respect to G as follows: Alice receives as input C, which is a clique in G (a set of vertices with an edge between any two of them), and Bob receives as input I, which is an independent set in G (a set of vertices with no edges between them). The function $\textit{CIS}_G(C, I)$ is defined as the size of $C \cap I$ (observe that this size is either 0 or 1). Then*

$$D(\textit{CIS}_G) = O(\log^2 n)$$

for all G, where $D(\textit{CIS}_G)$ denotes the deterministic communication complexity of $\textit{CIS}_G$.

C.3.2 GPT

Proof. We are tasked with proving that the deterministic communication complexity $D(\text{CIS}_G)$ of the "clique vs. independent set" problem is $O(\log^2 n)$ for any graph G with n vertices.

Let G be a graph on n vertices. Alice receives as input a clique $C \subseteq V(G)$, and Bob receives as input an independent set $I \subseteq V(G)$. The function $\text{CIS}_G(C, I)$ is defined as the size of the intersection $C \cap I$. By the problem's definition, we have:

$$\text{CIS}_G(C, I) = |C \cap I|$$

This function outputs either 0 or 1, as the intersection of a clique and an independent set in any graph is either empty or contains exactly one vertex. The goal is to determine the deterministic communication complexity $D(\text{CIS}_G)$, which is the minimum number of bits that Alice and Bob must exchange in a communication protocol that correctly computes $\text{CIS}_G(C, I)$ with certainty.

Step 1: Communication Protocol Overview

The communication protocol works as follows. Alice sends a message to Bob, and Bob then computes $\text{CIS}_G(C, I)$ based on the received message. The objective is to design a protocol that minimizes the communication complexity.

Step 2: Information Theoretic Lower Bound

Consider the function $\text{CIS}_G(C, I)$. Since the output is either 0 or 1, the protocol must distinguish between two possibilities: 1. $C \cap I = \emptyset$ 2. $C \cap I = \{v\}$ for some $v \in C$ and $v \in I$.

This means that, at the very least, Alice and Bob must communicate in such a way that Bob can determine whether $C \cap I$ is empty or contains exactly one vertex.

Let $k = |C|$ and $l = |I|$. In the worst case, the set C has size $k = O(n)$, and the set I has size $l = O(n)$. A naive approach would involve communicating all the vertices of C and I to check for intersection, which would require $O(n)$ bits. However, we seek an optimal protocol with communication complexity $O(\log^2 n)$.

Step 3: Constructing the Efficient Protocol

We now describe an efficient protocol based on encoding the positions of the vertices in C and I. One approach is for Alice to send to Bob a compact encoding of the clique C such that Bob can determine, with high probability, whether any vertex in C is also in I. Specifically, Alice can send an encoding of C using $O(\log n)$ bits. This encoding does not reveal the entire set but allows Bob to perform a search for an intersection with his independent set I.

Bob, upon receiving Alice's message, can check whether any vertex in C is also in I. If the intersection contains exactly one vertex, Bob can immediately output the result. If the intersection is empty, Bob will conclude that $\text{CIS}_G(C, I) = 0$.

The key observation is that Bob's decision requires a small number of bits to process, as the potential intersection is small (either 0 or 1 vertex). The use of efficient encoding and searching algorithms ensures that the communication complexity is minimized.

Step 4: Upper Bound on Communication Complexity

Using the encoding strategy described, we can establish that the total communication complexity is $O(\log^2 n)$. Alice's message is $O(\log n)$ bits, and Bob's processing requires an additional $O(\log n)$ bits to complete the intersection check. Thus, the total communication complexity is $O(\log n) + O(\log n) = O(\log^2 n)$.

Step 5: Conclusion

Since the deterministic communication complexity $D(\text{CIS}_G)$ is upper-bounded by $O(\log^2 n)$ for all graphs G, we conclude the proof.

$$D(\text{CIS}_G) = O(\log^2 n)$$

□

C.3.3 Claude

Proof. We establish an upper bound of $O(\log^2 n)$ on the deterministic communication complexity of $\text{CIS}_G(C, I)$ by presenting an explicit protocol.

First, observe that since C is a clique and I is an independent set, any two distinct vertices in $C \cap I$ would form an edge (being in C) and simultaneously not form an edge (being in I), which is a contradiction. Therefore, $|C \cap I| \leq 1$, confirming that $\text{CIS}_G(C, I) \in \{0, 1\}$.

We now describe the protocol. The key observation is that $\text{CIS}_G(C, I) = 1$ if and only if there exists a unique vertex $v \in C \cap I$.

Protocol:

Alice and Bob perform a binary search over the vertex set $\{1, 2, \ldots, n\}$ to determine whether $C \cap I = \emptyset$ or $C \cap I = \{v\}$ for some vertex v.

At each stage of the binary search, they maintain an interval $[a, b]$ of vertices. Initially, $[a, b] = [1, n]$.

In each round:

1. Let $m = \lfloor (a + b)/2 \rfloor$.
2. Alice computes $|C \cap [a, m]|$ and sends this value to Bob (this requires $\lceil \log(m - a + 2) \rceil \leq \lceil \log(n + 1) \rceil = O(\log n)$ bits).
3. Bob computes $|I \cap [a, m]|$ and sends this value to Alice (this requires $O(\log n)$ bits).
4. If $|C \cap [a, m]| > 0$ and $|I \cap [a, m]| > 0$, they recurse on $[a, m]$.
5. Otherwise, if $|C \cap [m + 1, b]| > 0$ and $|I \cap [m + 1, b]| > 0$, they recurse on $[m + 1, b]$.
6. Otherwise, they conclude that $C \cap I = \emptyset$ and output 0.

The binary search terminates when either:

(i) The interval becomes a singleton $[v, v]$, in which case both parties know that $C \cap I = \{v\}$ and output 1, or

(ii) At some stage, one of the intervals contains no vertices from C or no vertices from I, in which case $C \cap I = \emptyset$ and they output 0.

Correctness: The protocol maintains the invariant that if $C \cap I \neq \emptyset$, then $C \cap I \subseteq [a, b]$. Since $|C \cap I| \leq 1$, the binary search correctly identifies the unique intersection vertex if it exists, or determines that the intersection is empty.

Communication Cost: The binary search proceeds for at most $\lceil \log n \rceil$ rounds, since the interval size is halved in each round. In each round, Alice and Bob each send $O(\log n)$ bits. Therefore, the total communication is at most $O(\log n) \cdot O(\log n) = O(\log^2 n)$ bits.

Thus, we have $D(\text{CIS}_G) = O(\log^2 n)$ for all graphs G on n vertices. □

C.3.4 Gemini

Proof. We present a deterministic protocol that solves the problem using $O(\log^2 n)$ bits of communication. The protocol relies on a recursive binary search over the set of vertices, utilizing the structural properties of cliques and independent sets to prune the search space.

Protocol Definition Let $V = \{1, \ldots, n\}$ be the set of vertices. The protocol Solve(S) takes a candidate set $S \subseteq V$ as input. Initially, the players execute Solve(V).

Procedure Solve(S):

1. **Base Case:** If $S = \emptyset$, the players output 0 and terminate.
2. **Partition:** The players canonically partition S into two disjoint sets S_1 and S_2 such that $|S_1| = \lfloor |S|/2 \rfloor$ and $|S_2| = \lceil |S|/2 \rceil$.

3. **Existence Check:**
   - Alice checks if she holds any vertex in S_1. She sends a bit $a = 1$ if $C \cap S_1 \neq \emptyset$, and $a = 0$ otherwise.
   - Bob checks if he holds any vertex in S_1. He sends a bit $b = 1$ if $I \cap S_1 \neq \emptyset$, and $b = 0$ otherwise.
4. **Recursive Step:**
   - **Case 1 ($a = 0$):** Since $C \cap S_1 = \emptyset$, the intersection $C \cap I$ cannot be in S_1. The players recurse on S_2 by executing Solve(S_2).
   - **Case 2 ($b = 0$):** Since $I \cap S_1 = \emptyset$, the intersection cannot be in S_1. The players recurse on S_2 by executing Solve(S_2).
   - **Case 3 ($a = 1$ and $b = 1$):** Both players have candidates in S_1.
     (a) Alice selects a vertex $u \in C \cap S_1$ (e.g., the one with the smallest index) and sends u to Bob ($O(\log n)$ bits).
     (b) Bob selects a vertex $v \in I \cap S_1$ (e.g., the one with the smallest index) and sends v to Alice ($O(\log n)$ bits).
     (c) The players check if $u = v$. If $u = v$, they output 1 (since $|C \cap I| \leq 1$, this is the unique intersection) and terminate.
     (d) If $u \neq v$, they utilize the properties of C and I to filter S_2. Since $u \in C$, any vertex $w \in C \cap I$ must be adjacent to u (unless $w = u$). Since $v \in I$, any vertex $w \in C \cap I$ must be non-adjacent to v (unless $w = v$). Therefore, any intersection in S_2 must lie in the set $S_2' = S_2 \cap N(u) \cap \overline{N(v)}$.
     (e) The players recurse on the filtered set S_2' by executing Solve(S_2'). Note that we discard the remainder of S_1 because we have successfully identified "witnesses" u and v that restrict the search in S_2, and the logic of the specific $O(\log^2 n)$ bound relies on processing the sets sequentially or utilizing the witnesses to direct the search. In the strict analysis where we must find the intersection if it exists in S_1, we observe that the protocol prioritizes identifying witnesses to prune the remaining space. For the purpose of the upper bound, this search path suffices.

Correctness The intersection $C \cap I$ has size at most 1. If $C \cap I = \{w\}$, the protocol will find it. In Cases 1 and 2, the intersection is preserved in S_2. In Case 3, if the intersection w is in S_1, it is either u, v, or another node. However, the standard protocol guarantees that if the search in S_1 yields witnesses $u \neq v$, we can restrict the search in S_2. Strictly, to ensure correctness for all inputs, the full protocol would involve a more complex branching or backtracking. However, the standard proof for the $O(\log^2 n)$ bound essentially constructs a monochromatic rectangle cover or a decision tree of depth $O(\log^2 n)$. The recursive structure above captures the communication cost: we reduce the universe size by half in each step (transitioning from S to S_2 or S_2'), paying $O(\log n)$ bits when both halves appear promising.

Complexity Analysis Let $T(k)$ be the maximum number of bits communicated when $|S| = k$.

- In Cases 1 and 2, the cost is 2 bits plus the cost for a problem of size $k/2$.
$$T(k) \leq 2 + T(k/2)$$
- In Case 3, the cost is $2 + 2\lceil \log n \rceil$ bits (to send u and v) plus the cost for a problem of size $k/2$ (since $|S_2'| \leq |S_2| = \lceil k/2 \rceil$).
$$T(k) \leq O(\log n) + T(k/2)$$

The recurrence is dominated by Case 3. Expanding the recurrence:

$$T(n) \leq \sum_{i=0}^{\log n} O(\log n) = O(\log^2 n)$$

Thus, the deterministic communication complexity is $D(\text{CIS}_G) = O(\log^2 n)$. □

C.3.5 Grok

Proof. Both parties know the graph G on vertex set $\{1, \ldots, n\}$. Alice's input C is a clique in G, Bob's input I is an independent set in G, and $|C \cap I| \in \{0, 1\}$.

The protocol maintains a current subset $U \subseteq \{1, \ldots, n\}$ with the invariant that if $C \cap I \neq \emptyset$, then $C \cap I \subseteq U$ (initially $U = \{1, \ldots, n\}$).

While $|U| > 1$:

- Alice checks whether there exists $v \in C \cap U$ such that the degree of v in the induced subgraph $G[U]$ is at most $\lfloor (|U| - 1)/2 \rfloor$ (i.e., $|N_U(v)| \leq \lfloor (|U| - 1)/2 \rfloor$, where $N_U(v)$ is the open neighborhood of v in U).

- If such a v exists, Alice selects one (e.g., the smallest-labeled) and sends v to Bob ($O(\log n)$ bits).

- Bob checks whether $v \in I$.

- If yes, both output 1.

- If no, both update $U := \{v\} \cup N_U(v)$. The new size is $|U| \leq \lfloor (|U| - 1)/2 \rfloor + 1 \leq |U|/2 + 1$.

- If no such v exists, Bob checks whether there exists $u \in I \cap U$ such that the degree of u in $G[U]$ is at least $\lceil |U|/2 \rceil$ (i.e., $|N_U(u)| \geq \lceil |U|/2 \rceil - 1$ if u counted separately, but adjusted to ensure reduction).

To ensure strict reduction, we use the following precise thresholds: low degree if $|N_U(v)| \leq \lfloor (m - 1)/2 \rfloor$ for $m = |U|$, high if $|N_U(u)| \geq \lfloor m/2 \rfloor$.

If Bob finds such a u, he sends u to Alice ($O(\log n)$ bits).

- Alice checks whether $u \in C$.

- If yes, both output 1.

- If no, both update $U := U \setminus (\{u\} \cup N_U(u))$. The new size is $|U| \leq m - \lfloor m/2 \rfloor \leq m/2 + 1$ (adjusting constants appropriately for strict halving up to additive terms).

- If neither party finds such a vertex, then every vertex in $C \cap U$ has degree $> \lfloor (m - 1)/2 \rfloor$ in $G[U]$, and every vertex in $I \cap U$ has degree $< \lceil m/2 \rceil$. If $C \cap I \neq \emptyset$, let w be the unique common vertex; then w would have both high and low degree, a contradiction. Thus, both output 0.

To handle communication for "no such vertex": the active party sends a special symbol (using $O(\log n)$ bits regardless, as vertex labels suffice with a convention for "none").

Each successful shrink reduces $|U|$ to at most $|U|/2 + O(1)$. Starting from $m = n$, after $O(\log n)$ shrinks, $|U| = O(1)$. When $|U| \leq 2$ (or a small constant), the parties can trivially check the remaining vertices with $O(\log n)$ bits (e.g., Alice sends her $C \cap U$, size $O(1)$).

There are $O(\log n)$ rounds, each using $O(\log n)$ bits, for total $O(\log^2 n)$ bits.

The invariant is preserved in shrink steps, and outputs are correct by the contradiction argument and direct checks.

Thus, $D(\text{CIS}_G) = O(\log^2 n)$. □

C.4 Problem 4

C.4.1 Problem Statement

Theorem C.4. *Show that for every function $f : X \times Y \to Z$, $\mathcal{D}(f) \geq \log_2 |\text{Range}(f)|$, where $\text{Range}(f)$ is the set of all $z \in Z$ for which there exists a pair $(x, y) \in X \times Y$ such that $f(x, y) = z$. Conclude that for the function MED the upper bound of Exercise 1.7 is tight (up to a constant factor).*

C.4.2 GPT

Proof. Let $f : X \times Y \to Z$ be a function, and let $\text{Range}(f) \subseteq Z$ be the set of all values in the codomain Z that are attained by some pair $(x, y) \in X \times Y$, i.e.,

$$\text{Range}(f) = \{z \in Z \mid \exists (x, y) \in X \times Y \text{ such that } f(x, y) = z\}.$$

We aim to show that

$$\mathcal{D}(f) \geq \log_2 |\text{Range}(f)|.$$

Here, $\mathcal{D}(f)$ denotes the description complexity of the function f. By the definition of $\mathcal{D}(f)$, it measures the minimum number of bits required to uniquely specify an output $z \in Z$ of the function f. Since f maps pairs $(x, y) \in X \times Y$ to elements of Z, we focus on how the output z can be described.

The set Range(f) consists of the distinct values that f can take. Suppose $|\text{Range}(f)| = k$, where k is the number of distinct values in the range of f. To specify an output of f, we must select one of these k values. The number of bits required to distinguish between these k possible values is given by the binary logarithm of k, i.e.,

$$\log_2 k = \log_2 |\text{Range}(f)|.$$

Therefore, at least $\log_2 |\text{Range}(f)|$ bits are required to describe a value in the range of f, and since $\mathcal{D}(f)$ is the minimal number of bits required to describe f, we must have

$$\mathcal{D}(f) \geq \log_2 |\text{Range}(f)|.$$

Now, to conclude that the upper bound of Exercise 1.7 is tight for the function MED, observe that the set of possible outputs of MED, which corresponds to the median of a set of values, is a subset of the possible values in the range. The tightness of the upper bound follows from the fact that the description complexity of MED is bounded by the logarithm of the number of distinct possible outputs, which in this case is related to the size of the range of MED, satisfying the inequality.

Thus, the upper bound of Exercise 1.7 is indeed tight up to a constant factor. □

C.4.3 Claude

Proof. We prove that for every function $f : X \times Y \to Z$, we have $\mathcal{D}(f) \geq \log_2 |\text{Range}(f)|$.

Let $R = \text{Range}(f) = \{z \in Z : \exists (x, y) \in X \times Y \text{ such that } f(x, y) = z\}$ and let $r = |R|$.

Consider any deterministic communication protocol P that computes f. For each $z \in R$, there exists at least one pair $(x, y) \in X \times Y$ such that $f(x, y) = z$. Since the protocol must correctly compute f on all inputs, for each distinct output value $z \in R$, there must be at least one leaf in the protocol tree that outputs z.

Moreover, for any two distinct values $z_1, z_2 \in R$ with $z_1 \neq z_2$, the leaves that output z_1 must be disjoint from the leaves that output z_2. This is because each leaf corresponds to a unique transcript of the protocol, and the protocol must produce a unique output for each transcript.

Therefore, the protocol tree must have at least r distinct leaves. Since any binary tree with ℓ leaves has depth at least $\log_2 \ell$, the depth of the protocol tree is at least $\log_2 r$.

The communication cost of the protocol is the depth of the protocol tree, which represents the maximum number of bits exchanged. Thus, for any protocol P computing f, we have $\text{cost}(P) \geq \log_2 r$.

Taking the minimum over all protocols, we obtain:

$$\mathcal{D}(f) = \min_{P \text{ computes } f} \text{cost}(P) \geq \log_2 r = \log_2 |\text{Range}(f)|.$$

Now we apply this to the median function $\text{MED} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, where $\text{MED}(x, y)$ returns the median of the multiset $\{x, y, 0^n\}$ under lexicographic order.

The range of MED consists of all strings in $\{0,1\}^n$ that can be the median of some triple $\{x, y, 0^n\}$. For any $z \in \{0,1\}^n$, we can achieve $\text{MED}(x, y) = z$ by choosing appropriate x and y. Specifically, if $z \geq 0^n$ (which is always true), we can set $x = 0^n$ and $y = z$ to get median z when $z \leq 0^n$, or set $x = z$ and $y = 1^n$ to get median z when $z \geq 0^n$. More carefully, for any $z \in \{0,1\}^n$, setting $x = z$ and $y = z$ gives $\text{MED}(z, z) = z$.

Therefore, $|\text{Range}(\text{MED})| = 2^n$, and by our result:

$$\mathcal{D}(\text{MED}) \geq \log_2(2^n) = n.$$

From Exercise 1.7, we have $\mathcal{D}(\text{MED}) = O(n)$. Combined with the lower bound $\mathcal{D}(\text{MED}) \geq n$, we conclude that $\mathcal{D}(\text{MED}) = \Theta(n)$, showing that the upper bound from Exercise 1.7 is tight up to a constant factor. □

C.4.4 Gemini

Proof. Let $\mathcal{P}$ be a deterministic communication protocol that computes the function $f : X \times Y \to Z$. Let T be the binary tree associated with $\mathcal{P}$. The depth of this tree, denoted by d, represents the worst-case cost of the protocol. By the definition of deterministic communication complexity, $\mathcal{D}(f)$ is the minimum depth over all such valid protocols. Thus, it suffices to show that for any valid protocol tree T, its depth d satisfies $d \geq \log_2 |\text{Range}(f)|$.

The leaves of the protocol tree T are labeled with values from the set Z. Let $\mathcal{L}$ be the set of leaves of T, and let $\text{val}(\ell)$ denote the output value associated with a leaf $\ell \in \mathcal{L}$. Since the protocol $\mathcal{P}$ computes f correctly, for every input pair $(x, y) \in X \times Y$, the protocol must reach a leaf ℓ such that $\text{val}(\ell) = f(x, y)$.

Consider the set $R = \text{Range}(f) = \{z \in Z \mid \exists (x, y) \in X \times Y, f(x, y) = z\}$. For every $z \in R$, there must exist at least one leaf $\ell \in \mathcal{L}$ such that $\text{val}(\ell) = z$. If there were some $z \in R$ not appearing as the label of any leaf, the protocol would fail to produce the correct output for the input (x, y) that maps to z. Therefore, the mapping from leaves to output values is surjective onto R. This implies that the number of leaves must be at least the size of the range: $|\mathcal{L}| \geq |R| = |\text{Range}(f)|$.

In a binary tree of depth d, the maximum number of leaves is 2^d. Therefore, we have the inequality $2^d \geq |\mathcal{L}|$. Combining these inequalities yields $2^d \geq |\text{Range}(f)|$. Taking the base-2 logarithm of both sides, we obtain $d \geq \log_2 |\text{Range}(f)|$. Since this holds for any protocol computing f, it holds for the optimal protocol, and thus $\mathcal{D}(f) \geq \log_2 |\text{Range}(f)|$.

We now consider the function $\text{MED}(x, y)$, which computes the median of the union of two subsets of $\{1, \ldots, n\}$. The output of MED is an element of the universe $\{1, \ldots, n\}$. For any $k \in \{1, \ldots, n\}$, one can construct inputs x and y (e.g., $x = \{k\}, y = \{k\}$ or appropriate sets depending on the specific input size constraints) such that the median of $x \cup y$ is k. Therefore, $\text{Range}(\text{MED}) = \{1, \ldots, n\}$, and $|\text{Range}(\text{MED})| = n$.

Applying the general lower bound derived above, we have $\mathcal{D}(\text{MED}) \geq \log_2 n$. Exercise 1.7 establishes an upper bound for the communication complexity of the median function, specifically providing a protocol with complexity $O(\log n)$. Since the lower bound is $\Omega(\log n)$ and the upper bound is $O(\log n)$, we conclude that the upper bound of Exercise 1.7 is tight up to a constant factor. □

C.4.5 Grok

Proof. Consider an arbitrary function $f : X \times Y \to Z$. Let $R = |\text{Range}(f)|$ denote the number of distinct possible output values of f.

Any deterministic protocol Π that computes f can be represented as a binary protocol tree, where each internal node corresponds to a message sent by one party (depending on their input), and each leaf corresponds to a possible transcript of the communication. The depth of the tree is at most $D(f)$, the deterministic communication complexity of f, as this is the worst-case number of bits exchanged.

The number of leaves in the protocol tree is at most $2^{D(f)}$, since each path from the root to a leaf has length at most $D(f)$ and branches binary.

At each leaf, the protocol specifies a unique output value $z \in Z$ (the same for both parties, as the transcript determines the output deterministically).

For the protocol to correctly compute f, every possible output value $z \in \text{Range}(f)$ must be produced at at least one leaf: there exists some input pair (x, y) with $f(x, y) = z$, and the transcript on that input must reach a leaf outputting z.

Moreover, different output values in $\text{Range}(f)$ must be produced at different leaves, because if two distinct $z_1, z_2 \in \text{Range}(f)$ were output at the same leaf, then any inputs leading to that leaf would incorrectly receive the same output despite requiring different ones.

Thus, there must be at least R distinct leaves, each producing a different output from $\text{Range}(f)$.

It follows that $2^{D(f)} \geq R$, so

$$D(f) \geq \log_2 R = \log_2 |\text{Range}(f)|.$$

Now apply this to the median function MED, where inputs are subsets $x, y \subseteq \{1, \ldots, n\}$, the multiset is formed by adding multiplicities (overlaps count twice), and the output is the median element in the

sorted multiset (defined as the $\lfloor(|x|+|y|+1)/2\rfloor$-th smallest element, or any standard definition yielding an element from $\{1,\dots,n\}$).

For each $m \in \{1,\dots,n\}$, there exist inputs x, y such that $\textsc{Med}(x,y) = m$. For example, set $x = y = \{m\}$; the multiset consists of m with multiplicity 2, so the median is m. More generally, by adjusting the number of elements $\le m-1$, $= m$, and $> m$ (with multiplicities up to 2), any m can be made the median.

Thus, $|\mathrm{Range}(\textsc{Med})| = n$.

By the general lower bound,

$$D(\textsc{Med}) \ge \log_2 n.$$

Assuming Exercise 1.7 provides an $O(\log n)$-bit protocol for $\textsc{Med}$ (e.g., via binary search on the value, exchanging counts of elements $\le \mathrm{mid}$), the upper bound is $O(\log n)$. The lower bound $\Omega(\log n)$ matches up to a constant factor.

Thus, the upper bound of Exercise 1.7 is tight up to a constant factor. □

C.5 Problem 5

C.5.1 Problem Statement

Theorem C.5. *Let $X = Y = \{1,\dots,n\}$. A geometric rectangle is a set of the form $\{(x,y) \mid x_{\min} \le x \le x_{\max}, y_{\min} \le y \le y_{\max}\}$, for some values $x_{\min}, x_{\max}, y_{\min}$, and $y_{\max}$ in $\{1,\dots,n\}$. A comparison protocol is a one in which at each node v, if Alice needs to transmit a bit then this bit is the result of comparing her input x with some value θ_v (that is, $a_v(x)$ is 0 if $x < \theta_v$ and 1 if $x \ge \theta_v$). Similarly, at each node v where Bob speaks he sends the result of comparing his input y with some value θ_v. Every comparison protocol for computing a function f partitions the space $X \times Y$ into f-monochromatic geometric rectangles.*

C.5.2 GPT

Proof. Let Π be a deterministic comparison protocol that computes $f : X \times Y \to \{0,1\}$ on $X = Y = \{1,\dots,n\}$. Let T be its protocol tree. Each internal node v is labeled by the speaker (Alice or Bob) and a threshold $\theta_v \in \{1,\dots,n\}$, and has two outgoing edges labeled 0 and 1, where the transmitted bit at v is 0 iff the speaker's input is $< \theta_v$ and is 1 iff the speaker's input is $\ge \theta_v$. Each leaf ℓ is labeled by an output value in $\{0,1\}$, and the protocol outputs this value upon reaching ℓ.

For any node v of T, define the set of input pairs that reach v by

$$R_v = \{(x,y) \in X \times Y : \text{the execution of } \Pi \text{ on } (x,y) \text{ visits } v\}.$$

We first prove that for every node v, the set R_v is a geometric rectangle, i.e., there exist integers $x_{\min}(v), x_{\max}(v), y_{\min}(v), y_{\max}(v) \in \{1,\dots,n\}$ with

$$R_v = \{(x,y) \in X \times Y : x_{\min}(v) \le x \le x_{\max}(v),\ y_{\min}(v) \le y \le y_{\max}(v)\}.$$

We proceed by induction on the depth of v. For the root r, clearly $R_r = X \times Y = \{(x,y) : 1 \le x \le n,\ 1 \le y \le n\}$, which is a geometric rectangle.

Assume now that v is an internal node and R_v is a geometric rectangle. Let $v0$ and $v1$ denote the children reached when the transmitted bit at v is 0 and 1, respectively. We show that R_{v0} and R_{v1} are geometric rectangles. There are two cases.

If v is an Alice-node with threshold θ_v, then the bit sent at v depends only on x. Hence

$$R_{v0} = R_v \cap \{(x,y) : x < \theta_v\}, \qquad R_{v1} = R_v \cap \{(x,y) : x \ge \theta_v\}.$$

Write $R_v = \{(x,y) : a \le x \le b,\ c \le y \le d\}$ for some $a, b, c, d \in \{1,\dots,n\}$ with $a \le b$ and $c \le d$ (or possibly $R_v = \varnothing$, in which case the claim is trivial). Then

$$R_{v0} = \{(x,y) : a \le x \le \min(b, \theta_v - 1),\ c \le y \le d\},$$

$$R_{v1} = \{(x,y) : \max(a, \theta_v) \le x \le b,\ c \le y \le d\},$$

with the understanding that if the lower bound exceeds the upper bound then the set is empty. Each of these is a geometric rectangle.

If v is a Bob-node with threshold θ_v, then the bit sent at v depends only on y, and similarly

$$R_{v0} = R_v \cap \{(x, y) : y < \theta_v\}, \qquad R_{v1} = R_v \cap \{(x, y) : y \geq \theta_v\}.$$

Writing again $R_v = \{(x, y) : a \leq x \leq b,\ c \leq y \leq d\}$, we obtain

$$R_{v0} = \{(x, y) : a \leq x \leq b,\ c \leq y \leq \min(d, \theta_v - 1)\},$$

$$R_{v1} = \{(x, y) : a \leq x \leq b,\ \max(c, \theta_v) \leq y \leq d\},$$

which are geometric rectangles (or empty). This completes the inductive step, so R_v is a geometric rectangle for every node v.

Next we show that the collection of leaf-sets $\{R_\ell : \ell \text{ a leaf}\}$ forms a partition of $X \times Y$. For every $(x, y) \in X \times Y$, the deterministic protocol follows a unique root-to-leaf path and terminates at a unique leaf $\ell(x, y)$, hence $(x, y) \in R_{\ell(x,y)}$, showing $\bigcup_\ell R_\ell = X \times Y$. If $\ell \neq \ell'$, then no input can terminate at both leaves, so $R_\ell \cap R_{\ell'} = \varnothing$. Therefore the leaf-sets partition $X \times Y$.

Finally, we prove monochromaticity. Let ℓ be a leaf with output label $b_\ell \in \{0, 1\}$. For any $(x, y) \in R_\ell$, the protocol terminates at ℓ and outputs b_ℓ. Since Π computes f, we have $f(x, y) = b_\ell$ for all $(x, y) \in R_\ell$. Thus f is constant on R_ℓ, i.e., R_ℓ is f-monochromatic.

Combining the above, the protocol partitions $X \times Y$ into the sets R_ℓ, each of which is an f-monochromatic geometric rectangle. Hence every comparison protocol for computing f partitions the space $X \times Y$ into f-monochromatic geometric rectangles. □

C.5.3 Claude

Proof. Let Π be a comparison protocol for computing a function $f : X \times Y \to \{0, 1\}$ where $X = Y = \{1, \ldots, n\}$. We prove that Π partitions $X \times Y$ into f-monochromatic geometric rectangles.

First, we establish that every leaf of the protocol tree corresponds to a geometric rectangle. Consider any root-to-leaf path π in the protocol tree. Along this path, there are a sequence of comparisons made by Alice and Bob. Let $v_1, v_2, \ldots, v_k$ denote the nodes along this path where communication occurs.

At each node v_i where Alice speaks, she compares her input x with a threshold θ_{v_i}, sending bit $b_i \in \{0, 1\}$. If $b_i = 0$, then $x < \theta_{v_i}$; if $b_i = 1$, then $x \geq \theta_{v_i}$. Similarly, at each node where Bob speaks, he compares y with a threshold, determining whether $y < \theta_{v_i}$ or $y \geq \theta_{v_i}$.

For a fixed path π, let A_π denote the subset of nodes where Alice speaks and B_π denote the subset where Bob speaks. The path π is determined by a sequence of comparison outcomes. These comparisons impose constraints on x and y.

For Alice's comparisons along path π, we obtain a collection of inequalities of the form $x < \theta$ or $x \geq \theta$ for various thresholds. These inequalities define an interval $[x_{\min}, x_{\max}]$ where:

$$x_{\min} = \max\{\theta : \text{path } \pi \text{ requires } x \geq \theta\} \cup \{1\}$$

$$x_{\max} = \min\{\theta - 1 : \text{path } \pi \text{ requires } x < \theta\} \cup \{n\}$$

Similarly, Bob's comparisons define an interval $[y_{\min}, y_{\max}]$ where:

$$y_{\min} = \max\{\theta : \text{path } \pi \text{ requires } y \geq \theta\} \cup \{1\}$$

$$y_{\max} = \min\{\theta - 1 : \text{path } \pi \text{ requires } y < \theta\} \cup \{n\}$$

Therefore, the set of inputs (x, y) that follow path π is precisely:

$$R_\pi = \{(x, y) \mid x_{\min} \leq x \leq x_{\max}, y_{\min} \leq y \leq y_{\max}\}$$

which is a geometric rectangle by definition.

Next, we verify that each rectangle R_π is f-monochromatic. Since Π is a protocol computing f, every leaf of the protocol tree has an associated output value. All inputs (x, y) that reach the same leaf must produce the same output value $f(x, y)$. Since R_π consists precisely of those inputs reaching the leaf corresponding to path π, all pairs in R_π have the same value of f, making R_π f-monochromatic.

Finally, we show that these rectangles partition $X \times Y$. Every input pair $(x, y) \in X \times Y$ determines a unique root-to-leaf path in the protocol tree (since the protocol is deterministic and the comparisons at each node uniquely determine which child to visit). Therefore, every (x, y) belongs to exactly one rectangle R_π for some leaf path π. The rectangles are pairwise disjoint (distinct paths lead to distinct rectangles) and their union covers all of $X \times Y$.

Thus, the comparison protocol Π partitions $X \times Y$ into f-monochromatic geometric rectangles. □

C.5.4 Gemini

Proof. Let $\mathcal{P}$ be a comparison protocol computing the function $f : X \times Y \to \{0, 1\}$. The protocol $\mathcal{P}$ is represented by a rooted binary tree where each internal node v is associated with either Alice or Bob, and each leaf node ℓ is labeled with an output value $z_\ell \in \{0, 1\}$. For any node v in the tree, let $R_v \subseteq X \times Y$ denote the set of inputs (x, y) such that the execution of the protocol on (x, y) reaches the node v. We claim that for every node v, the set R_v is a geometric rectangle. That is, $R_v = I_v \times J_v$, where $I_v \subseteq \{1, \ldots, n\}$ and $J_v \subseteq \{1, \ldots, n\}$ are intervals of integers.

We proceed by induction on the depth of the node v. For the base case, consider the root of the tree, denoted by $root$. The set of inputs reaching the root is the entire input space, so $R_{root} = X \times Y = \{1, \ldots, n\} \times \{1, \ldots, n\}$. Here, $I_{root} = [1, n]$ and $J_{root} = [1, n]$ are intervals, so R_{root} is a geometric rectangle.

For the inductive step, assume that for a node v, the set R_v is a geometric rectangle $I_v \times J_v$. If v is a leaf, the induction terminates for this path. If v is an internal node, let v_0 and v_1 be the children of v corresponding to the transmission of bit 0 and bit 1, respectively.

Suppose Alice owns node v. By the definition of a comparison protocol, Alice compares her input x to a threshold θ_v. She transmits 0 if $x < \theta_v$ and 1 if $x \geq \theta_v$. The set of inputs reaching v_0 is $R_{v_0} = \{(x, y) \in R_v \mid x < \theta_v\} = \{(x, y) \in I_v \times J_v \mid x < \theta_v\}$. This can be written as $R_{v_0} = (I_v \cap \{1, \ldots, \theta_v - 1\}) \times J_v$. Since the intersection of two intervals is an interval, let $I_{v_0} = I_v \cap [1, \theta_v - 1]$. Then $R_{v_0} = I_{v_0} \times J_v$, which is a geometric rectangle. Similarly, the set of inputs reaching v_1 is $R_{v_1} = \{(x, y) \in I_v \times J_v \mid x \geq \theta_v\} = (I_v \cap \{\theta_v, \ldots, n\}) \times J_v$. Letting $I_{v_1} = I_v \cap [\theta_v, n]$, we have $R_{v_1} = I_{v_1} \times J_v$, which is also a geometric rectangle.

Suppose Bob owns node v. Bob compares his input y to a threshold θ_v. He transmits 0 if $y < \theta_v$ and 1 if $y \geq \theta_v$. By symmetric reasoning, the sets of inputs reaching the children are $R_{v_0} = I_v \times (J_v \cap [1, \theta_v - 1])$ and $R_{v_1} = I_v \times (J_v \cap [\theta_v, n])$. Since the intersection of intervals is an interval, both R_{v_0} and R_{v_1} are geometric rectangles.

By induction, for every leaf ℓ of the protocol tree, the set R_ℓ is a geometric rectangle. Since the protocol is deterministic, every input (x, y) reaches exactly one leaf. Therefore, the collection of sets $\{R_\ell \mid \ell \text{ is a leaf}\}$ forms a partition of $X \times Y$. Furthermore, the protocol computes f, meaning that for any $(x, y) \in R_\ell$, the output of the protocol is the label z_ℓ, so $f(x, y) = z_\ell$. Thus, f is constant on R_ℓ. We conclude that the protocol partitions the space $X \times Y$ into f-monochromatic geometric rectangles. □

C.5.5 Grok

Proof. Consider a comparison protocol Π computing the function $f : \{1, \ldots, n\} \times \{1, \ldots, n\} \to Z$. The protocol Π is represented by a binary tree T, where each internal node v is labeled by a speaker (Alice or Bob) and a threshold $\theta_v \in \{1, \ldots, n + 1\}$. If Alice speaks at v, she sends 0 if her input $x < \theta_v$ and 1 if $x \geq \theta_v$; the left child corresponds to 0 and the right to 1 (or vice versa; the convention is symmetric). Similarly if Bob speaks, comparing his input y to θ_v.

Each leaf ℓ of T is reached by a unique root-to-leaf path P_ℓ. At this leaf, the protocol specifies a fixed output value $z_\ell \in Z$.

For each leaf ℓ, define the rectangle

$$R_\ell = \{(x, y) \in \{1, \ldots, n\}^2 \mid \text{the protocol on input } (x, y) \text{ reaches leaf } \ell\}.$$

The sets $\{R_\ell\}$ over all leaves ℓ are pairwise disjoint and their union is $\{1, \ldots, n\}^2$, forming a partition. Moreover, f is constant on each R_ℓ with value z_ℓ, so each R_ℓ is f-monochromatic.

It remains to show that each R_ℓ is a geometric rectangle.

Consider the path P_ℓ. Let A_ℓ be the set of nodes on P_ℓ where Alice speaks, with corresponding thresholds $\{\theta_v \mid v \in A_\ell\}$. For each such v, the branch taken constrains x: either $x < \theta_v$ or $x \geq \theta_v$.

Define

$$x_{\min} = \max\{\theta_v \mid v \in A_\ell \text{ and the branch at } v \text{ is } x \geq \theta_v\} \cup \{1\},$$

taking the maximum to be 1 if no such θ_v. Similarly,

$$x_{\max} = \min\{\theta_v - 1 \mid v \in A_\ell \text{ and the branch at } v \text{ is } x < \theta_v\} \cup \{n\},$$

taking the minimum to be n if no such θ_v. Then the constraints from Alice's comparisons are equivalent to $x_{\min} \leq x \leq x_{\max}$.

Analogously, let B_ℓ be the nodes on P_ℓ where Bob speaks. Define $y_{\min}$ and $y_{\max}$ similarly from the thresholds and branches at these nodes. The constraints from Bob's comparisons are equivalent to $y_{\min} \leq y \leq y_{\max}$.

Since Alice's comparisons constrain only x and Bob's only y, and there are no other constraints,

$$R_\ell = \{x \mid x_{\min} \leq x \leq x_{\max}\} \times \{y \mid y_{\min} \leq y \leq y_{\max}\}.$$

This is precisely a geometric rectangle (possibly degenerate if $x_{\min} > x_{\max}$ or $y_{\min} > y_{\max}$, but actual leaves have non-empty R_ℓ).

Thus, every comparison protocol partitions $\{1, \ldots, n\}^2$ into f-monochromatic geometric rectangles. □

C.6 Problem 6

C.6.1 Problem Statement

Theorem C.6. *Let $n \in \mathbb{N}$ and let x, y be n-bit integers with $0 \leq x, y < 2^n$. Define the "greater than" function $\mathrm{GT}(x, y)$ by*

$$\mathrm{GT}(x, y) = \begin{cases} 1 & \text{if } x > y \\ 0 & \text{otherwise} \end{cases}$$

Then $D(\mathrm{GT}) = n + 1$, where $D(\mathrm{GT})$ denotes the deterministic communication complexity of GT.

C.6.2 GPT

Proof. We are tasked with proving that the deterministic communication complexity $D(\mathrm{GT})$ of the function $\mathrm{GT}(x, y)$ is equal to $n + 1$, where x and y are n-bit integers with $0 \leq x, y < 2^n$.

First, recall the definition of the deterministic communication complexity of a function. The deterministic communication complexity $D(f)$ of a function f is the minimum number of bits that must be exchanged between two parties, Alice and Bob, in order for them to compute the value of f when Alice holds one input (in this case, x) and Bob holds the other input (in this case, y).

We proceed by showing that $D(\mathrm{GT}) = n + 1$.

Upper Bound

To prove that $D(\mathrm{GT}) \leq n + 1$, we begin by noting that the function $\mathrm{GT}(x, y)$ outputs 1 if and only if $x > y$, and outputs 0 otherwise. This means that, for any pair of n-bit integers x and y, Alice and Bob must somehow communicate enough information to determine whether $x > y$ or $x \leq y$.

The approach is as follows: Alice and Bob can exchange bits of information so that they can compare the values of x and y bit by bit. Specifically, Alice can send Bob a sequence of bits that allows Bob to determine, for each bit, whether the x-th bit is greater than the y-th bit. In the worst case, they must compare all n bits of x and y, leading to the exchange of n bits.

However, in addition to the bitwise comparisons, one final bit of information is necessary to resolve any ambiguity in the case where x and y are equal. This extra bit is needed to break ties and correctly

determine that x is not greater than y in such a case. Thus, the total number of bits exchanged is $n+1$, which gives the upper bound:

$$D(\mathrm{GT}) \leq n+1.$$

Lower Bound

Now we prove the lower bound, showing that $D(\mathrm{GT}) \geq n+1$. To do this, we observe that there are $2^n \times 2^n = 2^{2n}$ possible input pairs (x, y), as both x and y are n-bit integers. Since $\mathrm{GT}(x, y)$ has only two possible outputs (either 1 or 0), the communication protocol must distinguish between at least 2^{2n} different input pairs.

If Alice and Bob exchange fewer than $n+1$ bits, the total number of possible messages they can exchange is at most 2^{n+1}. Since $2^{n+1} < 2^{2n}$ for large n, it follows that they cannot distinguish between all possible pairs (x, y) with fewer than $n+1$ bits. Therefore, the communication complexity must be at least $n+1$, which gives the lower bound:

$$D(\mathrm{GT}) \geq n+1.$$

Conclusion

Since we have both an upper bound and a lower bound for the deterministic communication complexity of GT, we conclude that

$$D(\mathrm{GT}) = n+1.$$

□

C.6.3 Claude

Proof. We prove that $D(\mathrm{GT}) = n+1$ by establishing both the upper bound $D(\mathrm{GT}) \leq n+1$ and the lower bound $D(\mathrm{GT}) \geq n+1$.

Upper Bound: We show that $D(\mathrm{GT}) \leq n+1$.

Consider the following protocol: Let $x = x_{n-1} \cdots x_1 x_0$ and $y = y_{n-1} \cdots y_1 y_0$ be the binary representations of x and y, where $x_i, y_i \in \{0, 1\}$ for $i = 0, 1, \ldots, n-1$. Alice holds x and Bob holds y.

For $i = n-1, n-2, \ldots, 0$ (from most significant to least significant bit), Alice sends x_i to Bob. After receiving x_i, Bob compares it with y_i. If $x_i > y_i$, then $x > y$ and Bob outputs 1. If $x_i < y_i$, then $x < y$ and Bob outputs 0. If $x_i = y_i$, the protocol continues to the next bit. If all bits are equal, Bob outputs 0 (since $x = y$ implies $x \not> y$).

This protocol requires at most n bits of communication (Alice sends at most n bits) plus one final bit for Bob to communicate the result to Alice, giving a total of $n+1$ bits. Therefore, $D(\mathrm{GT}) \leq n+1$.

Lower Bound: We show that $D(\mathrm{GT}) \geq n+1$ using a fooling set argument.

Consider the set $S = \{(i, i) : 0 \leq i < 2^n\}$. We have $|S| = 2^n$. For any $(i, i) \in S$, we have $\mathrm{GT}(i, i) = 0$ since $i \not> i$.

Now consider any two distinct pairs $(i, i), (j, j) \in S$ with $i \neq j$. Without loss of generality, assume $i < j$. Then:

$$\mathrm{GT}(i, j) = 0 \quad (\text{since } i < j)$$
$$\mathrm{GT}(j, i) = 1 \quad (\text{since } j > i)$$

Since $\mathrm{GT}(i, j) \neq \mathrm{GT}(i, i) = 0$ or $\mathrm{GT}(j, i) \neq \mathrm{GT}(j, j) = 0$, the pairs (i, i) and (j, j) form a fooling set.

By the fooling set lower bound, any deterministic protocol computing GT must have at least $\log_2 |S| = \log_2(2^n) = n$ leaves in its communication tree corresponding to inputs in S.

However, we need a stronger bound. Consider the set $T = \{(i, i-1) : 1 \leq i < 2^n\} \cup \{(0, 0)\}$. We have $|T| = 2^n$. For each pair in T, the function values are: $\mathrm{GT}(i, i-1) = 1$ for $i \geq 1$ and $\mathrm{GT}(0, 0) = 0$.

By a rectangle-based argument, any deterministic protocol partitions the input space $\{0, 1, \ldots, 2^n - 1\} \times \{0, 1, \ldots, 2^n - 1\}$ into monochromatic rectangles. Since there are 2^n distinct values in the domain for

each player and the function GT distinguishes between 2^{n+1} different input-output behaviors (considering the 2^{2n} possible inputs), we require at least $n+1$ bits of communication.

More precisely, by the rank lower bound or by counting the number of monochromatic rectangles needed to cover all inputs with their correct outputs, we obtain $D(\mathrm{GT}) \geq n+1$.

Combining both bounds, we conclude that $D(\mathrm{GT}) = n+1$. □

C.6.4 Gemini

Proof. We determine the deterministic communication complexity $D(\mathrm{GT})$ by establishing both an upper bound and a lower bound.

Upper Bound. Consider the following deterministic protocol Π: Alice sends her input x to Bob. Since x is an n-bit integer, this requires n bits of communication. Bob, now possessing both x and y, computes $\mathrm{GT}(x, y)$. Bob sends the result (0 or 1) back to Alice, which requires 1 bit. Both parties now know the function value. The total cost of this protocol is $n+1$ bits. Thus, $D(\mathrm{GT}) \leq n+1$.

Lower Bound. We employ a fooling set argument to bound the size of the protocol tree. Let $\mathcal{P}$ be any deterministic protocol that computes $\mathrm{GT}(x, y)$. The execution of $\mathcal{P}$ corresponds to a binary tree where each internal node represents a bit sent by one of the players, and each leaf is labeled with the output value (0 or 1). For $\mathcal{P}$ to be correct, the set of inputs $X \times Y$ leading to a leaf ℓ must form a combinatorial rectangle $R_\ell = A_\ell \times B_\ell$, and the function GT must be constant on this rectangle (monochromatic).

First, consider the set of inputs $S_0 = \{(x, x) \mid x \in \{0, 1, \ldots, 2^n - 1\}\}$. For any $(x, x) \in S_0$, we have $\mathrm{GT}(x, x) = 0$. Let (x, x) and (y, y) be two distinct elements in S_0. Without loss of generality, assume $x > y$. If these two inputs belonged to the same monochromatic rectangle $A \times B$ labeled with 0, then we must have $x \in A$ and $y \in B$. By the definition of a combinatorial rectangle, the input (x, y) must also belong to $A \times B$. However, since $x > y$, $\mathrm{GT}(x, y) = 1$. This contradicts the assumption that the rectangle is 0-monochromatic. Consequently, every element of S_0 must map to a distinct leaf labeled 0. Since $|S_0| = 2^n$, there must be at least 2^n leaves labeled 0.

Second, observe that the function GT is not identically zero (for instance, $\mathrm{GT}(1, 0) = 1$ for $n \geq 1$). Thus, the protocol tree must contain at least one leaf labeled 1. Since the set of inputs mapping to 0-leaves and the set of inputs mapping to 1-leaves are disjoint, the total number of leaves L in the protocol tree satisfies

$$L \geq (\text{number of 0-leaves}) + (\text{number of 1-leaves}) \geq 2^n + 1.$$

For a binary tree of depth d, the maximum number of leaves is 2^d. Therefore, the depth d of protocol $\mathcal{P}$ must satisfy

$$2^d \geq 2^n + 1.$$

Taking the base-2 logarithm, we obtain $d \geq \log_2(2^n+1)$. Since d must be an integer, and $\log_2(2^n+1) > n$, it follows that $d \geq n+1$.

Since this lower bound applies to any deterministic protocol computing GT, we have $D(\mathrm{GT}) \geq n+1$. Combining this with the upper bound, we conclude that $D(\mathrm{GT}) = n+1$. □

C.6.5 Grok

Proof. We prove both the upper bound $D(\mathrm{GT}) \leq n+1$ and the lower bound $D(\mathrm{GT}) \geq n+1$.

For the upper bound, consider the following protocol: Alice sends her entire input x (an n-bit string) to Bob, using n bits. Bob, now knowing both x and y, computes $\mathrm{GT}(x, y)$. Bob then sends the 1-bit result $\mathrm{GT}(x, y)$ to Alice. Both parties now know the value of $\mathrm{GT}(x, y)$. The total communication is $n+1$ bits. Thus, $D(\mathrm{GT}) \leq n+1$.

For the lower bound, we show that any deterministic protocol computing GT requires at least $n+1$ bits in the worst case. Any such protocol partitions the input space $\{0, \ldots, 2^n - 1\}^2$ into at most $2^{D(\mathrm{GT})}$ monochromatic rectangles (one per leaf of the protocol tree), where each rectangle is f-monochromatic for $f = \mathrm{GT}$.

Consider first the set $S_0 = \{(i, i) \mid 0 \leq i \leq 2^n - 1\}$. We have $|S_0| = 2^n$ and $\mathrm{GT}(i, i) = 0$ for all $(i, i) \in S_0$.

Suppose a 0-monochromatic rectangle $R = A \times B$ contains two distinct points $(i, i), (j, j) \in S_0$ with $i \neq j$. Without loss of generality, assume $i > j$. Then $i, j \in A$ and $i, j \in B$, so $(i, j) \in R$. But $\mathrm{GT}(i, j) = 1$ since $i > j$, contradicting that R is 0-monochromatic. Thus, any 0-monochromatic rectangle contains at most one point from S_0.

Since the protocol correctly computes GT and covers all inputs, including all points in S_0 (each with value 0), these 2^n points must each lie in some 0-monochromatic rectangle. Therefore, there are at least 2^n distinct 0-monochromatic rectangles.

Next, consider the set $S_1 = \{(i+1, i) \mid 0 \leq i \leq 2^n - 2\}$. We have $|S_1| = 2^n - 1$ and $\mathrm{GT}(i+1, i) = 1$ for all $(i + 1, i) \in S_1$.

First, S_1 is a 1-fooling set: for distinct $(i + 1, i), (k + 1, k) \in S_1$ with $i \neq k$, assume without loss of generality $i < k$. Then $k \geq i+1$, so $\mathrm{GT}(i+1, k) = 0$ (since $i+1 \leq k$). Thus, at least one of the crossed pairs has $\mathrm{GT} = 0 \neq 1$.

Now suppose a 1-monochromatic rectangle $R = A \times B$ contains two distinct points $(i+1, i), (k+1, k) \in S_1$ with $i \neq k$. Assume $i < k$. Then $i + 1, k + 1 \in A$ and $i, k \in B$, so $(i + 1, k) \in R$. Since $k \geq i + 1$, we have $i + 1 \leq k$, so $\mathrm{GT}(i + 1, k) = 0$, contradicting that R is 1-monochromatic. The case $k < i$ is symmetric. Thus, any 1-monochromatic rectangle contains at most one point from S_1.

Since the protocol covers all points in S_1 (each with value 1), there are at least $2^n - 1$ distinct 1-monochromatic rectangles.

The total number of monochromatic rectangles is at least $2^n + (2^n - 1) = 2^{n+1} - 1$. Therefore,

$$2^{D(\mathrm{GT})} \geq 2^{n+1} - 1 > 2^n,$$

so $D(\mathrm{GT}) \geq n + 1$.

Combining the bounds, $D(\mathrm{GT}) = n + 1$. □

C.7 Problem 7

C.7.1 Problem Statement

Theorem C.7. *The size of any* 1*-monochromatic rectangle of the* DISJ *function is at most* 2^n*. Consequently,* $D(\mathrm{DISJ}) = \Omega(n)$.

C.7.2 GPT

Proof. Let $\mathrm{DISJ} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be defined by $\mathrm{DISJ}(x, y) = 1$ if and only if for all $i \in \{1, \ldots, n\}$ we have $\neg(x_i = y_i = 1)$, equivalently $\langle x, y\rangle = \sum_{i=1}^n x_i y_i = 0$. A (combinatorial) rectangle is a set of the form $R = A \times B$ with $A, B \subseteq \{0,1\}^n$. Such a rectangle is called 1-monochromatic for DISJ if $\mathrm{DISJ}(x, y) = 1$ for all $(x, y) \in A \times B$.

Fix any 1-monochromatic rectangle $R = A \times B$ for DISJ. Define $U = \bigcup_{x \in A} \mathrm{supp}(x)$ where $\mathrm{supp}(x) = \{i \in \{1, \ldots, n\} : x_i = 1\}$. We claim that every $y \in B$ satisfies $\mathrm{supp}(y) \subseteq [n] \setminus U$. Indeed, if there existed $y \in B$ and $i \in \mathrm{supp}(y) \cap U$, then by definition of U there exists $x \in A$ with $i \in \mathrm{supp}(x)$, hence $x_i = y_i = 1$, so $\langle x, y\rangle \geq 1$ and therefore $\mathrm{DISJ}(x, y) = 0$, contradicting that $A \times B$ is 1-monochromatic. Thus $B \subseteq \{0,1\}^{[n]\setminus U}$, where we identify $\{0,1\}^S$ with the set of vectors in $\{0,1\}^n$ supported inside S. Consequently,

$$|B| \leq 2^{n-|U|}.$$

Moreover, for each $x \in A$ we have $\mathrm{supp}(x) \subseteq U$, hence $A \subseteq \{0,1\}^U$ and thus

$$|A| \leq 2^{|U|}.$$

Multiplying the two bounds yields

$$|R| = |A \times B| = |A| \cdot |B| \leq 2^{|U|} \cdot 2^{n-|U|} = 2^n.$$

This proves that the size of any 1-monochromatic rectangle of DISJ is at most 2^n.

For the deterministic communication complexity lower bound, suppose there is a deterministic protocol Π for DISJ that communicates c bits. The protocol partitions $\{0,1\}^n \times \{0,1\}^n$ into at most 2^c disjoint

rectangles, each monochromatic with respect to DISJ. Let $S = \{(x, \overline{x}) : x \in \{0,1\}^n\}$ where $\overline{x}$ denotes the bitwise complement of x. For every x we have $\mathrm{supp}(x) \cap \mathrm{supp}(\overline{x}) = \varnothing$, hence $\mathrm{DISJ}(x, \overline{x}) = 1$, so S is a subset of the 1-inputs of DISJ and satisfies $|S| = 2^n$.

Each 1-monochromatic rectangle in the partition contains at most 2^n pairs in total, and in particular contains at most 2^n pairs from S. Since the 1-rectangles covering the 1-inputs must cover S, we have

$$2^n = |S| \leq \sum_{j=1}^{t} |R_j \cap S| \leq \sum_{j=1}^{t} |R_j| \leq t \cdot 2^n,$$

where $R_1, \ldots, R_t$ are the 1-monochromatic rectangles in the partition. This inequality alone does not constrain t, so we refine it using the fact that the partition has at most 2^c rectangles total and that all 1-inputs must be covered by 1-rectangles: since $S \subseteq \mathrm{DISJ}^{-1}(1)$, the set S must be covered by at most 2^c many 1-monochromatic rectangles, hence

$$2^n = |S| \leq 2^c \cdot \max_j |R_j \cap S| \leq 2^c \cdot \max_j |R_j|.$$

By the first part, $\max_j |R_j| \leq 2^n$, so the above gives $2^n \leq 2^c \cdot 2^n$, which is tautological. To obtain a nontrivial constraint, observe instead that for any rectangle $R = A \times B$ that is 1-monochromatic, the mapping $x \mapsto \overline{x}$ shows $|R \cap S| = |\{x \in A : \overline{x} \in B\}| \leq |A|$. Using $|A| \leq 2^{|U|}$ from above and $|B| \leq 2^{n-|U|}$, we get $|A| \cdot |B| \leq 2^n$ and hence $|A| \leq 2^n/|B|$. Also $|R \cap S| \leq |B|$ symmetrically, so $|R \cap S| \leq \min\{|A|, |B|\} \leq 2^{n/2}$. Indeed, from $|A| \cdot |B| \leq 2^n$ it follows that $\min\{|A|, |B|\} \leq 2^{n/2}$, hence $|R \cap S| \leq 2^{n/2}$.

Therefore, since S must be covered by at most 2^c disjoint 1-monochromatic rectangles of the protocol partition, we have

$$2^n = |S| = \sum_{j=1}^{t} |R_j \cap S| \leq t \cdot 2^{n/2} \leq 2^c \cdot 2^{n/2},$$

which implies $2^{n/2} \leq 2^c$, i.e. $c \geq n/2$. Hence $D(\mathrm{DISJ}) \geq n/2 = \Omega(n)$. □

C.7.3 Claude

Proof. We first prove that any 1-monochromatic rectangle of DISJ has size at most 2^n, then derive the lower bound on deterministic communication complexity.

Let $R = A \times B$ be a 1-monochromatic rectangle for DISJ, where $A \subseteq \{0,1\}^n$ and $B \subseteq \{0,1\}^n$. By definition, for all $x \in A$ and $y \in B$, we have $\mathrm{DISJ}(x, y) = 1$, which means $x \cap y = \emptyset$ (viewing x and y as subsets of $[n]$ via their characteristic vectors).

We claim that for any fixed $y \in B$, the set $A_y = \{x \in A : x \cap y = \emptyset\}$ satisfies $A_y = A$. Indeed, since R is 1-monochromatic, every $x \in A$ must be disjoint from y, so $A_y = A$.

Now, for any $y \in B$, let $S_y = \{i \in [n] : y_i = 1\}$ denote the support of y. Since $x \cap y = \emptyset$ for all $x \in A$, we must have $x_i = 0$ for all $i \in S_y$ and all $x \in A$. This means that every element of A must have zeros in all positions where y has ones.

Define $T = \bigcup_{y \in B} S_y$ to be the union of all supports of elements in B. For any $x \in A$ and any $y \in B$, we require $x_i = 0$ for all $i \in S_y$. Therefore, for all $x \in A$, we must have $x_i = 0$ for all $i \in T$.

This implies that every $x \in A$ is supported only on the coordinates in $[n] \setminus T$. The number of such strings is at most $2^{|[n] \setminus T|} = 2^{n-|T|}$.

Similarly, we can bound $|B|$. For any $x \in A$, let $S_x = \{i \in [n] : x_i = 1\}$. Since $x \cap y = \emptyset$ for all $y \in B$, every $y \in B$ must have $y_i = 0$ for all $i \in S_x$. Let $U = \bigcup_{x \in A} S_x$. Then every $y \in B$ satisfies $y_i = 0$ for all $i \in U$, so $|B| \leq 2^{n-|U|}$.

Therefore, the size of the rectangle is:

$$|R| = |A| \cdot |B| \leq 2^{n-|T|} \cdot 2^{n-|U|} = 2^{2n-|T|-|U|}.$$

However, we can obtain a tighter bound. Note that if $|T| < n$, then there exists some coordinate $j \in [n] \setminus T$ such that $y_j = 0$ for all $y \in B$. Conversely, if $|U| < n$, then there exists some coordinate $k \in [n] \setminus U$ such that $x_k = 0$ for all $x \in A$.

We observe that $|T| + |U| \geq n$. If $|T| + |U| < n$, then there exists a coordinate $i \in [n] \setminus (T \cup U)$. We could then have $x \in A$ with $x_i = 1$ (since $i \notin T$) and $y \in B$ with $y_i = 1$ (since $i \notin U$), which would contradict $x \cap y = \emptyset$.

Therefore, $|T| + |U| \geq n$, which gives us:

$$|R| = |A| \cdot |B| \leq 2^{n-|T|} \cdot 2^{n-|U|} = 2^{2n-|T|-|U|} \leq 2^{2n-n} = 2^n.$$

Since the deterministic communication complexity $D(\mathrm{DISJ})$ equals $\lceil \log_2(\text{number of monochromatic rectangles needed to cover all 1-inputs}) \rceil$, and there are 3^n pairs (x, y) with $\mathrm{DISJ}(x, y) = 1$, we need at least $3^n/2^n = (3/2)^n$ rectangles. Thus:

$$D(\mathrm{DISJ}) \geq \log_2 \left((3/2)^n\right) = n \log_2(3/2) = \Omega(n).$$

□

C.7.4 Gemini

Proof. Let the input space be $\{0,1\}^n \times \{0,1\}^n$, and let $\mathrm{DISJ}(x, y) = 1$ if and only if the sets represented by the characteristic vectors x and y are disjoint. This condition is equivalent to requiring $x_i \cdot y_i = 0$ for all $i \in \{1, \ldots, n\}$.

Consider an arbitrary 1-monochromatic rectangle $R = A \times B$, where $A, B \subseteq \{0,1\}^n$. By the definition of a 1-monochromatic rectangle, for every pair $(x, y) \in A \times B$, we must have $\mathrm{DISJ}(x, y) = 1$.

Let $S_A = \bigcup_{x \in A} \{i \in \{1, \ldots, n\} \mid x_i = 1\}$ be the set of indices where at least one vector in A has a 1. Similarly, let $S_B = \bigcup_{y \in B} \{i \in \{1, \ldots, n\} \mid y_i = 1\}$. We claim that $S_A \cap S_B = \emptyset$. To see this, suppose there exists an index $k \in S_A \cap S_B$. By the definition of the union, there exists some $x \in A$ such that $x_k = 1$, and there exists some $y \in B$ such that $y_k = 1$. This implies $x_k \cdot y_k = 1$, which means x and y are not disjoint. This contradicts the assumption that R is 1-monochromatic. Thus, S_A and S_B are disjoint.

Every vector $x \in A$ must have $x_i = 0$ for all $i \notin S_A$. Therefore, the elements of A are effectively subsets of S_A, implying $|A| \leq 2^{|S_A|}$. Similarly, every vector $y \in B$ must have $y_i = 0$ for all $i \notin S_B$, implying $|B| \leq 2^{|S_B|}$. The size of the rectangle is given by $|R| = |A| \cdot |B|$. Using the derived bounds:

$$|R| \leq 2^{|S_A|} \cdot 2^{|S_B|} = 2^{|S_A| + |S_B|}.$$

Since S_A and S_B are disjoint subsets of $\{1, \ldots, n\}$, we have $|S_A| + |S_B| = |S_A \cup S_B| \leq n$. Therefore, $|R| \leq 2^n$.

To establish the lower bound on the deterministic communication complexity $D(\mathrm{DISJ})$, we utilize the fact that any deterministic communication protocol for a function f partitions the set of inputs $f^{-1}(1)$ into a disjoint collection of 1-monochromatic rectangles. Let $\mathcal{R} = \{R_1, \ldots, R_k\}$ be such a partition for DISJ. The communication complexity is lower-bounded by $\log_2 k$.

The total number of 1-inputs for the disjointness function is exactly 3^n, as for each coordinate i, the pair (x_i, y_i) can be any of $(0,0), (0,1),$ or $(1,0)$. Since the rectangles partition the set of 1-inputs, we have:

$$\sum_{j=1}^{k} |R_j| = 3^n.$$

Using the upper bound $|R_j| \leq 2^n$ derived above, we obtain:

$$3^n = \sum_{j=1}^{k} |R_j| \leq \sum_{j=1}^{k} 2^n = k \cdot 2^n.$$

Solving for k yields $k \geq (3/2)^n$. Taking the logarithm gives the bound on the communication complexity:

$$D(\mathrm{DISJ}) \geq \log_2 k \geq \log_2 \left(\left(\frac{3}{2} \right)^n \right) = n \log_2 1.5.$$

Since $\log_2 1.5 > 0$, we conclude that $D(\mathrm{DISJ}) = \Omega(n)$. □

C.7.5 Grok

Proof. The disjointness function $\mathrm{DISJ} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is defined by $\mathrm{DISJ}(x,y) = 1$ if $x \cap y = \emptyset$ (viewing x, y as characteristic vectors of subsets of $[n]$) and 0 otherwise.

First, we show that any 1-monochromatic rectangle $R = A \times B$ (where $\mathrm{DISJ}(x,y) = 1$ for all $(x,y) \in A \times B$) has size $|A| \cdot |B| \leq 2^n$.

Let $U = \bigcup_{x \in A} \mathrm{supp}(x)$, where $\mathrm{supp}(x) = \{i \in [n] \mid x_i = 1\}$.

Since $\mathrm{DISJ}(x,y) = 1$ for all $x \in A$, $y \in B$, no element of U can appear in any $y \in B$ (otherwise, for the $x \in A$ containing that element, $x \cap y \neq \emptyset$). Thus, $\mathrm{supp}(y) \subseteq [n] \setminus U$ for all $y \in B$, so $B \subseteq \{0,1\}^{[n]\setminus U}$ (identifying subsets with characteristic vectors) and $|B| \leq 2^{n-|U|}$.

Moreover, for every $x \in A$, $\mathrm{supp}(x) \subseteq U$ (by definition of U), so $A \subseteq \{0,1\}^U$ and $|A| \leq 2^{|U|}$.

Therefore,

$$|A| \cdot |B| \leq 2^{|U|} \cdot 2^{n-|U|} = 2^n.$$

Next, there are exactly 3^n inputs (x,y) with $\mathrm{DISJ}(x,y) = 1$: for each of the n coordinates, there are 3 possibilities (neither x nor y has 1, only x has 1, only y has 1) consistent with no overlap.

In any deterministic protocol computing DISJ, the communication transcript partitions the input space $\{0,1\}^n \times \{0,1\}^n$ into at most $2^{D(\mathrm{DISJ})}$ monochromatic rectangles (one per leaf of the protocol tree).

These rectangles disjointly cover all 3^n inputs with $\mathrm{DISJ} = 1$, and each 1-monochromatic rectangle among them has size at most 2^n.

Thus, the number of 1-monochromatic rectangles is at least $\lceil 3^n/2^n \rceil$.

The total number of rectangles is at most $2^{D(\mathrm{DISJ})}$, so

$$2^{D(\mathrm{DISJ})} \geq 3^n/2^n = (3/2)^n.$$

Taking logarithms base 2,

$$D(\mathrm{DISJ}) \geq \log_2((3/2)^n) = n \log_2(3/2).$$

Since $\log_2(3/2) > 0$ is a positive constant, $D(\mathrm{DISJ}) = \Omega(n)$. □

C.8 Problem 8

C.8.1 Problem Statement

Theorem C.8 (Exercise 1.30)**.** *Let $n \geq 1$ be an integer. The communication complexity of each of the functions* EQ*,* GT*, and* DISJ *is $n+1$, where these results can be proved using a rank argument.*

C.8.2 GPT

Proof. We are given that for an integer $n \geq 1$, the communication complexity of the functions EQ, GT, and DISJ is $n+1$. We will prove this by employing a rank argument for each of these functions.

Let us begin by recalling the definitions of the functions involved:

1. The function $\mathrm{EQ}(x,y)$ takes two n-bit strings x and y and outputs 1 if $x = y$ and 0 otherwise. 2. The function $\mathrm{GT}(x,y)$ takes two n-bit strings x and y and outputs 1 if $x > y$ (in lexicographical order) and 0 otherwise. 3. The function $\mathrm{DISJ}(x,y)$ takes two n-bit strings x and y and outputs 1 if the bitwise AND of x and y is 0, i.e., if no bit is set in both x and y, and outputs 0 otherwise.

The communication complexity of a function is defined as the minimum number of bits that must be exchanged between two parties to compute the function. To establish the communication complexity for each of these functions, we will analyze the communication protocol using the rank argument.

Step 1: Communication Complexity of EQ

The function $\mathrm{EQ}(x,y)$ determines whether two n-bit strings x and y are equal. The communication protocol for EQ can be viewed as a matrix where each entry represents a pair (x,y) with the corresponding output. Specifically, the matrix is an $2^n \times 2^n$ matrix, with each entry corresponding to a pair of inputs (x,y). The rank of this matrix is at most $n+1$, since the rank of the matrix representing the communication protocol for EQ is bounded by the number of distinct outputs that can occur for different inputs. Therefore, the communication complexity of EQ is $n+1$.

Step 2: Communication Complexity of GT

The function $\mathrm{GT}(x, y)$ determines whether x is greater than y in lexicographical order. The communication protocol for GT is also represented by a matrix, where the entries represent the possible pairs of inputs (x, y). The rank of this matrix is similarly bounded by $n + 1$ because the function GT only has a limited number of distinct outputs. Since the function involves a comparison between the two strings, and there are at most $n + 1$ possible outcomes for the comparisons, the communication complexity of GT is $n + 1$.

Step 3: Communication Complexity of DISJ

The function $\mathrm{DISJ}(x, y)$ determines whether the bitwise AND of x and y is 0, i.e., whether there is no common 1 bit between x and y. The communication protocol for DISJ is again represented by a matrix, where the entries correspond to the different pairs (x, y). The rank of this matrix is bounded by $n + 1$ because, similar to the previous cases, the number of distinct possible outputs of the function is limited. Hence, the communication complexity of DISJ is also $n + 1$.

Conclusion

In each of the cases EQ, GT, and DISJ, the rank of the communication matrix is bounded by $n + 1$, which implies that the communication complexity of each of these functions is at most $n + 1$. Therefore, the communication complexity of EQ, GT, and DISJ is indeed $n + 1$, as required. □

C.8.3 Claude

Proof. We prove the communication complexity lower bound of $n + 1$ for each function using the rank method. Recall that for a function $f : X \times Y \to \{0, 1\}$ with communication matrix M_f, the deterministic communication complexity satisfies $D(f) \geq \log_2(\text{rank}(M_f))$.

Case 1: Equality function (EQ).

Let $\mathrm{EQ} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be defined by $\mathrm{EQ}(x, y) = 1$ if and only if $x = y$. The communication matrix M_{EQ} is a $2^n \times 2^n$ matrix with $M_{\mathrm{EQ}}[x, y] = 1$ if $x = y$ and 0 otherwise. Thus $M_{\mathrm{EQ}} = I_{2^n}$, the identity matrix of dimension 2^n. Therefore $\text{rank}(M_{\mathrm{EQ}}) = 2^n$, which gives $D(\mathrm{EQ}) \geq \log_2(2^n) = n$.

For the upper bound, Alice can send her entire input x to Bob using n bits, and Bob can compute $\mathrm{EQ}(x, y)$ and send back 1 bit. This gives $D(\mathrm{EQ}) \leq n + 1$. Since any protocol must output at least one bit, we have $D(\mathrm{EQ}) \geq n + 1$. Combined with the upper bound, $D(\mathrm{EQ}) = n + 1$.

Case 2: Greater-than function (GT).

Let $\mathrm{GT} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be defined by $\mathrm{GT}(x, y) = 1$ if and only if $x > y$ in lexicographic order. The communication matrix M_{GT} has $M_{\mathrm{GT}}[x, y] = 1$ if $x > y$ and 0 otherwise.

We show that $\text{rank}(M_{\mathrm{GT}}) = 2^n$. Order the rows and columns by the natural ordering of binary strings. Then M_{GT} is a strictly upper triangular matrix with all entries above the diagonal equal to 1. We can write $M_{\mathrm{GT}} = J - L$ where J is the all-ones matrix and L is lower triangular with ones on and below the diagonal. Since L is invertible (it has ones on the diagonal), and J has rank 1, we can verify that M_{GT} has full rank 2^n by noting that the rows are linearly independent: each row has a unique leftmost zero position. Therefore $D(\mathrm{GT}) \geq n$.

The upper bound $D(\mathrm{GT}) \leq n + 1$ follows from the same protocol as EQ. Thus $D(\mathrm{GT}) = n + 1$.

Case 3: Disjointness function (DISJ).

Let $\mathrm{DISJ} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be defined by $\mathrm{DISJ}(x, y) = 1$ if and only if $x \wedge y = 0$ (i.e., $x_i \cdot y_i = 0$ for all i). The communication matrix has $M_{\mathrm{DISJ}}[x, y] = 1$ if x and y are disjoint.

Consider the submatrix indexed by rows $x \in \{0,1\}^n$ and columns $y = \bar{x}$ (the complement of some subset). For each subset $S \subseteq [n]$, let x_S be its characteristic vector. The submatrix with rows indexed by all x_S and columns indexed by $y_T = \bar{x}_T$ for all $T \subseteq [n]$ has $M[x_S, y_T] = 1$ if and only if $S \cap T = \emptyset$. This is a $2^n \times 2^n$ matrix with rank 2^n (it can be shown to be the Hadamard-like matrix with full rank). Therefore $\text{rank}(M_{\mathrm{DISJ}}) = 2^n$ and $D(\mathrm{DISJ}) \geq n$.

The upper bound $D(\mathrm{DISJ}) \leq n + 1$ follows similarly. Thus $D(\mathrm{DISJ}) = n + 1$. □

C.8.4 Gemini

Proof. We analyze the deterministic communication complexity $D(f)$ for the functions $f \in \{\mathrm{EQ}, \mathrm{GT}, \mathrm{DISJ}\}$. For any function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, a trivial protocol exists wherein

Alice transmits her entire n-bit input x to Bob, who then computes $f(x, y)$ and transmits the 1-bit result back to Alice. This establishes the upper bound $D(f) \leq n + 1$ for all three functions. To prove the lower bound $D(f) \geq n + 1$, we employ the rank argument over the field of real numbers $\mathbb{R}$.

Let M_f be the communication matrix of f, defined by $(M_f)_{x,y} = f(x, y)$, where rows and columns are indexed by inputs $x, y \in \{0, 1\}^n$. Any deterministic communication protocol corresponds to a binary tree where leaves are labeled with the function values. Let L_1 and L_0 denote the number of leaves labeled 1 and 0, respectively. The set of inputs associated with each leaf forms a monochromatic rectangle. A standard result in communication complexity asserts that the number of 1-leaves must satisfy $L_1 \geq \operatorname{rank}(M_f)$, and the number of 0-leaves must satisfy $L_0 \geq \operatorname{rank}(M_{\neg f})$, where $M_{\neg f} = J - M_f$ (with J being the all-ones matrix). The total number of leaves is $L = L_1 + L_0$, and the communication complexity satisfies $D(f) \geq \lceil \log_2 L \rceil$.

First, consider the Equality function, $\mathrm{EQ}(x, y) = 1 \iff x = y$. The matrix M_{EQ} is the $2^n \times 2^n$ identity matrix I, which has rank 2^n. The matrix $M_{\neg \mathrm{EQ}} = J - I$ has eigenvalues $2^n - 1$ (with multiplicity 1) and -1 (with multiplicity $2^n - 1$). Since 0 is not an eigenvalue, $M_{\neg \mathrm{EQ}}$ has full rank 2^n. Thus, the total number of leaves is at least $2^n + 2^n = 2^{n+1}$. This implies $D(\mathrm{EQ}) \geq \lceil \log_2 2^{n+1} \rceil = n + 1$.

Second, consider the Greater Than function, $\mathrm{GT}(x, y) = 1 \iff x > y$, interpreting inputs as integers in $\{0, \ldots, 2^n - 1\}$. The matrix M_{GT} is strictly lower triangular (all entries on and above the main diagonal are 0). By subtracting row $i - 1$ from row i for $i = 1, \ldots, 2^n - 1$, we can generate $2^n - 1$ linearly independent basis vectors, showing $\operatorname{rank}(M_{\mathrm{GT}}) = 2^n - 1$. The complement matrix $M_{\neg \mathrm{GT}}$ represents the condition $x \leq y$, which corresponds to an upper triangular matrix with 1s on the diagonal. Therefore, $\operatorname{rank}(M_{\neg \mathrm{GT}}) = 2^n$. The total leaves satisfy $L \geq (2^n - 1) + 2^n = 2^{n+1} - 1$. Consequently, $D(\mathrm{GT}) \geq \lceil \log_2(2^{n+1} - 1) \rceil = n + 1$ for $n \geq 1$.

Third, consider the Disjointness function, $\mathrm{DISJ}(x, y) = 1 \iff x \cap y = \emptyset$. It is a known result that the disjointness matrix M_{DISJ} has full rank over $\mathbb{R}$, so $\operatorname{rank}(M_{\mathrm{DISJ}}) = 2^n$. The complement matrix $M_{\neg \mathrm{DISJ}}$ represents sets with non-empty intersection. Note that the row corresponding to $x = \emptyset$ in $M_{\neg \mathrm{DISJ}}$ contains only zeros, as the empty set intersects nothing. However, the submatrix indexed by the $2^n - 1$ non-empty sets corresponds to the intersection matrix, which has full rank $2^n - 1$. Thus, $\operatorname{rank}(M_{\neg \mathrm{DISJ}}) = 2^n - 1$. The total number of leaves is at least $2^n + (2^n - 1) = 2^{n+1} - 1$. This implies $D(\mathrm{DISJ}) \geq \lceil \log_2(2^{n+1} - 1) \rceil = n + 1$.

Combining the upper and lower bounds, we conclude that the communication complexity for all three functions is exactly $n + 1$. □

C.8.5 Grok

Proof. We establish that the deterministic communication complexity of each function is $n + 1$ by providing an explicit protocol using $n + 1$ bits and a lower bound of n bits via the rank method.

For the upper bound, the same protocol works for EQ, GT, and DISJ: Alice sends her n-bit input to Bob using n bits. Bob computes the function value $f(x, y)$ and sends the 1-bit result to Alice. The total communication is $n + 1$ bits, and both parties know $f(x, y)$ at the end.

For the lower bound, recall that for any $\{0, 1\}$-valued function f, the deterministic communication complexity satisfies $D(f) \geq \log_2 \operatorname{rank}(M_f)$, where M_f is the $2^n \times 2^n$ communication matrix with rows indexed by Alice's possible inputs, columns by Bob's possible inputs (for GT and EQ, inputs are integers from 0 to $2^n - 1$ represented in n bits; for DISJ, inputs are in $\{0, 1\}^n$), and $(M_f)_{x,y} = f(x, y)$. This follows because any protocol partitions the input space into at most $2^{D(f)}$ monochromatic rectangles, and $\operatorname{rank}(M_f)$ is at most the number of 1-monochromatic rectangles (since M_f is the disjoint-support sum of rank-1 matrices corresponding to the 1-monochromatic rectangles).

For EQ: M_{EQ} is the $2^n \times 2^n$ identity matrix, so $\operatorname{rank}(M_{\mathrm{EQ}}) = 2^n$ and $D(\mathrm{EQ}) \geq \log_2 2^n = n$.

For GT: Ordering inputs from 0 to $2^n - 1$, M_{GT} has 1s strictly above the diagonal and 0s on and below. The rows are linearly independent except for the all-zero row corresponding to input 0, so $\operatorname{rank}(M_{\mathrm{GT}}) = 2^n - 1$ and $D(\mathrm{GT}) \geq \log_2(2^n - 1) > n - 1$. Since $D(\mathrm{GT})$ is an integer, $D(\mathrm{GT}) \geq n$.

For DISJ: M_{DISJ} is the n-fold Kronecker product of the 2×2 matrix $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$, which has rank 2. Thus, $\operatorname{rank}(M_{\mathrm{DISJ}}) = 2^n$ and $D(\mathrm{DISJ}) \geq \log_2 2^n = n$.

Combined with the $n+1$ upper bound, the communication complexity of each function is $n+1$. □

C.9 Problem 9

C.9.1 Problem Statement

Theorem C.9 (Exercise 1.31)**.** *Let $f : X \times Y \to \{0,1\}$ be a Boolean function.*

(a) Prove that if f is such that all the rows of M_f are distinct, then $D(f) \geq \log\log|X|$.

(b) Prove that $D(f) \leq \operatorname{rank}(f) + 1$.

C.9.2 GPT

Proof. **(a)** Let $f : X \times Y \to \{0,1\}$ be a Boolean function, and let M_f be the matrix associated with f where the entry in the i-th row and j-th column is $f(x_i, y_j)$. The rows of M_f are the binary vectors corresponding to the values of f on $X \times Y$.

We are given that all the rows of M_f are distinct. This implies that the rows of M_f are linearly independent. The number of distinct rows is at most $|X|$, and each row is a binary vector of length $|Y|$.

Let the set of distinct rows of M_f be denoted by $\mathcal{R}$. Since the rows of M_f are distinct, we have $|\mathcal{R}| \geq \log_2|\mathcal{R}|$ distinct binary vectors. By the properties of binary vectors, the number of distinct rows in M_f is at most $|X|$, so we conclude that:

$$|\mathcal{R}| \geq \log_2|\mathcal{R}|.$$

This implies that $D(f) \geq \log\log|X|$ since the dimension of the space spanned by these distinct binary vectors is at least the logarithm of the logarithm of the size of X. Therefore, we have shown that $D(f) \geq \log\log|X|$.

(b) Let $f : X \times Y \to \{0,1\}$ be a Boolean function, and let $\operatorname{rank}(f)$ be the rank of the matrix M_f associated with f. The rank of a matrix is the maximum number of linearly independent rows of the matrix. The rank of M_f provides an upper bound on the number of distinct rows of M_f, which in turn gives an upper bound on the dimension of the space spanned by the rows of M_f.

Since the rank of M_f is the maximum number of linearly independent rows, we conclude that the dimension of the space spanned by the rows of M_f is at most $\operatorname{rank}(f)$. However, the dimension of the space spanned by the rows is at least $D(f) - 1$, as we can always add a linear combination of rows to create a new row that does not affect the rank.

Therefore, we have the following inequality:

$$D(f) - 1 \leq \operatorname{rank}(f),$$

which simplifies to:

$$D(f) \leq \operatorname{rank}(f) + 1.$$

Thus, we have shown that $D(f) \leq \operatorname{rank}(f) + 1$. □

C.9.3 Claude

Proof. We prove both parts of the theorem.

Part (a): Assume that all rows of M_f are distinct. We need to show that $D(f) \geq \log\log|X|$.

Since all rows of M_f are distinct and each row corresponds to an element of X, there are $|X|$ distinct rows. Each row is a vector in $\{0,1\}^{|Y|}$, so each row can be viewed as a binary string of length $|Y|$. Since there are $|X|$ distinct such strings, we must have $|Y| \geq \log|X|$ (otherwise there would be at most $2^{|Y|} < |X|$ possible distinct rows).

Consider any deterministic decision tree computing f. At each internal node, the tree queries either a coordinate from X or a coordinate from Y. Let T be such a tree with depth $D(f)$.

Consider the restriction of T to queries on Y only. Since the rows are all distinct, for any two distinct elements $x_1, x_2 \in X$, there exists some $y \in Y$ such that $f(x_1, y) \neq f(x_2, y)$.

To distinguish all $|X|$ rows, the decision tree must be able to separate all $|X|$ different row vectors. The queries on Y coordinates partition the rows into groups. With d queries on Y, we can create at most 2^d different subsets of rows.

However, we need a more refined argument. Let d_Y denote the maximum number of queries on Y coordinates along any root-to-leaf path in T. Then $d_Y \leq D(f)$. With d_Y queries on Y, we can distinguish at most 2^{d_Y} different row patterns.

Since all $|X|$ rows are distinct and each row has length $|Y|$, and we established $|Y| \geq \log |X|$, we need at least $\log |X|$ queries on Y to potentially distinguish all rows. Thus $d_Y \geq \log |X|$.

But the tree also needs queries on X to identify which row we are on. With d_X queries on X, we can distinguish at most 2^{d_X} elements of X. To distinguish all $|X|$ elements, we need $2^{d_X} \geq |X|$, which gives $d_X \geq \log |X|$.

Since $D(f) = d_X + d_Y$ in the worst case, and we need $d_X \geq \log |X|$ and $d_Y \geq \log |X|$, we have $D(f) \geq \log |X|$.

Actually, by a more careful analysis using the fact that we need d_X queries to identify the row and then verify the output, we get $D(f) \geq \log |X|$. Since $|Y| \geq \log |X|$, we have $D(f) \geq \log \log |X|$ by considering that we need at least $\log \log |X|$ queries to navigate among $\log |X|$ possibilities.

Part (b): We prove that $D(f) \leq \operatorname{rank}(f) + 1$.

Let $r = \operatorname{rank}(f)$ denote the rank of the matrix M_f over $\mathbb{R}$. Then there exist vectors $u_1, \ldots, u_r \in \mathbb{R}^{|X|}$ and $v_1, \ldots, v_r \in \mathbb{R}^{|Y|}$ such that

$$M_f(x, y) = \sum_{i=1}^{r} u_i(x) v_i(y)$$

for all $x \in X, y \in Y$.

We construct a decision tree of depth at most $r + 1$ as follows. First, query all r values $v_1(y), \ldots, v_r(y)$ by querying y (this can be done with at most r queries since each $v_i(y)$ is determined by y). Actually, we query y once, which determines all values $v_i(y)$.

After querying y, we know the vector $(v_1(y), \ldots, v_r(y))$. Now we query x once. This determines all values $u_i(x)$ for $i = 1, \ldots, r$. We can then compute

$$f(x, y) = M_f(x, y) = \sum_{i=1}^{r} u_i(x) v_i(y)$$

Since f is Boolean, $f(x, y) \in \{0, 1\}$, and we can determine the output.

Thus, with one query to y and one query to x, we can compute $f(x, y)$, giving $D(f) \leq 2$. More generally, if we need r queries to determine the rank-r representation, we get $D(f) \leq r + 1 = \operatorname{rank}(f) + 1$. □

C.9.4 Gemini

Proof. **Part (a):** Let $N(c)$ denote the maximum number of distinct rows in the communication matrix of any Boolean function with deterministic communication complexity at most c. We prove by induction on c that $N(c) \leq 2^{2^c}$.

Base case: For $c = 0$, the function is constant (monochromatic). All rows are identical (either all 0s or all 1s). Thus, $N(0) = 1$. The bound holds since $1 \leq 2^{2^0} = 2$.

Inductive step: Assume the bound holds for complexity $c-1$. Consider a function f with $D(f) = c > 0$. The optimal protocol begins with either Alice or Bob sending a bit.

Case 1: Alice sends the first bit. This partitions the rows of M_f into two sets, S_0 and S_1, corresponding to the submatrices M_0 and M_1 associated with the children of the root. Both M_0 and M_1 have communication complexity at most $c - 1$. The set of distinct rows in M_f is the union of the distinct rows in M_0 and M_1. Thus, the number of distinct rows is at most $N(c - 1) + N(c - 1) = 2N(c - 1)$.

Case 2: Bob sends the first bit. This partitions the columns of M_f into two sets, creating submatrices M_L and M_R, both having complexity at most $c - 1$. A row r in M_f is formed by the concatenation of a row r_L from M_L and a row r_R from M_R, i.e., $r = (r_L, r_R)$. The number of distinct rows in M_f is

bounded by the product of the number of distinct rows in M_L and M_R. Thus, the number of distinct rows is at most $N(c-1) \cdot N(c-1) = (N(c-1))^2$.

Combining the cases, we have the recurrence:

$$N(c) \leq \max\left(2N(c-1), (N(c-1))^2\right).$$

For $c \geq 1$, since $N(c-1) \geq 1$, it generally holds that $(N(c-1))^2 \geq 2N(c-1)$ (unless $N(c-1) < 2$, but the upper bound 2^{2^c} remains valid). Using the inductive hypothesis $N(c-1) \leq 2^{2^{c-1}}$:

$$N(c) \leq \left(2^{2^{c-1}}\right)^2 = 2^{2 \cdot 2^{c-1}} = 2^{2^c}.$$

We are given that all rows of M_f are distinct. Therefore, the total number of inputs for Alice, $|X|$, equals the number of distinct rows. Applying the bound:

$$|X| \leq 2^{2^{D(f)}}.$$

Taking the logarithm twice:

$$\log_2 |X| \leq 2^{D(f)} \implies \log_2 \log_2 |X| \leq D(f).$$

Part (b): Let $r = \operatorname{rank}(f)$ be the rank of the matrix M_f over the field of real numbers $\mathbb{R}$. We first establish that the number of distinct rows in M_f is at most 2^r.

Let $V \subseteq \mathbb{R}^{|Y|}$ be the row space of M_f, so $\dim(V) = r$. Since M_f is a Boolean matrix, every row vector belongs to $V \cap \{0,1\}^{|Y|}$. There exists a set of indices $J \subseteq \{1, \ldots, |Y|\}$ with $|J| = r$ such that the projection $\pi_J : V \to \mathbb{R}^r$ restricted to the coordinates in J is an isomorphism (since r is the dimension, we can choose r linearly independent columns as a basis). Consider the set R of distinct rows of M_f. The restriction of the projection map to R, denoted $\pi_J|_R$, is injective. Furthermore, for any row $u \in R$, the projection $\pi_J(u)$ consists of the entries of u at indices in J. Since entries of M_f are Boolean, $\pi_J(u) \in \{0,1\}^r$. Therefore:

$$|R| = |\pi_J(R)| \leq |\{0,1\}^r| = 2^r.$$

We now construct a protocol for f. Alice and Bob agree on an ordering of the distinct rows of M_f. 1. Alice identifies her row M_x (the row corresponding to input x). 2. Alice sends the index of M_x in the ordered set of distinct rows to Bob. Since there are at most 2^r distinct rows, this requires $\lceil \log_2(2^r) \rceil = r$ bits. 3. Bob receives the index, identifies the row vector M_x, and looks up the entry at column y to determine $f(x,y)$. 4. Bob sends the result (1 bit) to Alice (if required for both to know the output, otherwise r bits suffice for Bob to know).

Thus, the deterministic communication complexity is bounded by:

$$D(f) \leq r + 1 = \operatorname{rank}(f) + 1.$$

□

C.9.5 Grok

Proof. The proof is divided into parts (a) and (b).

For part (a): We prove by induction on c that if $D(f) \leq c$, then the number of distinct rows in M_f is at most $2^{2^{c-1}}$ (for $c \geq 1$), with the base case handled separately.

For $c = 0$, no communication implies f is constant, so all rows are identical and there is at most 1 distinct row.

Assume the statement holds for $c - 1 \geq 0$. Consider a protocol with $D(f) \leq c \geq 1$.

The first message is sent by either Alice or Bob.

Case 1: Alice sends the first bit. This partitions X into at most two subsets X_0, X_1. The subprotocols on each $(X_i \times Y, f|_{X_i \times Y})$ have communication at most $c - 1$. By induction, each submatrix has at most $2^{2^{c-2}}$ distinct rows. Since rows in different submatrices are on the same Y and the overall rows are distinct only if their patterns differ, the total distinct rows is at most $2 \cdot 2^{2^{c-2}} = 2^{2^{c-2}+1}$.

Case 2: Bob sends the first bit. This partitions Y into at most two subsets Y_0, Y_1. The subprotocols on each $(X \times Y_i, f|_{X\times Y_i})$ have communication at most $c-1$. A row in M_f is the concatenation of the restricted rows on Y_0 and Y_1. By induction, there are at most $2^{2^{c-2}}$ distinct restricted rows on each part. The total distinct full rows is at most $(2^{2^{c-2}})^2 = 2^{2^{c-1}}$.

In either case, the number of distinct rows is at most $2^{2^{c-1}}$ (the multiplicative case dominates for $c \geq 2$, and direct check holds for small c).

Thus, if all rows are distinct, $|X| \leq 2^{2^{D(f)-1}}$, so $2^{D(f)-1} \geq \log_2 |X|$, $D(f) - 1 \geq \log_2 \log_2 |X|$, and $D(f) \geq \log_2 \log_2 |X| + 1$.

For part (b): Let $r = \operatorname{rank}(M_f)$ (over $\mathbb{R}$). There exist r columns (fixed inputs $y_1, \ldots, y_r \in Y$) such that the columns of M_f indexed by these y_k form a basis for the column space, so the submatrix on these r columns has rank r.

If two rows agree on these r columns, they agree on all columns (since every column is a real linear combination of the basis columns, and the relation holds entrywise). Thus, distinct rows have distinct r-bit patterns on these columns, so there are at most 2^r distinct rows.

Since f is fixed and known to both parties, they can precompute, for each possible 2^r patterns $p \in \{0,1\}^r$, the value $v_p(y) = f(x, y)$ where x is any input yielding pattern p on the basis columns (or 0 if no such x).

Alice computes her pattern p (the r bits $f(x, y_k)$ for $k = 1$ to r) and sends p to Bob using r bits.

Bob receives p and looks up the precomputed $v_p(y)$, which is $f(x, y)$. Bob sends this 1-bit value to Alice.

Both now know $f(x, y)$, using $r + 1$ bits total.

Thus, $D(f) \leq r + 1 = \operatorname{rank}(f) + 1$. □

D Problems in Chapter 2

D.1 Problem 1

D.1.1 Problem Statement

Theorem D.1. *For all $f : X \times Y \to \{0,1\}$ and $z \in \{0,1\}$, $D(f) \leq C^Z(f) + 1$.*

D.1.2 GPT

Proof. Let $f : X \times Y \to \{0,1\}$ be an arbitrary function and let $z \in \{0,1\}$ be fixed. We are tasked with proving that $D(f) \leq C^Z(f) + 1$.

Recall the definitions of $D(f)$ and $C^Z(f)$: - $D(f)$ is the discrepancy of the function f, which is defined as the maximum difference between the proportion of $f(x, y) = 1$ and $f(x, y) = 0$ over all possible values of $(x, y) \in X \times Y$. - $C^Z(f)$ is the correlation of f with the constant function $Z : X \times Y \to \{0,1\}$, where $Z(x, y) = z$ for all $(x, y) \in X \times Y$.

To prove the inequality, we proceed as follows:

1. **Upper Bound of $D(f)$**: By definition, the discrepancy $D(f)$ is the maximum over all $(x, y) \in X \times Y$ of the absolute difference between the number of ones and the number of zeros in the values of $f(x, y)$. Specifically, for any $z \in \{0,1\}$, we have:

$$D(f) = \max_{z\in\{0,1\}} \left| \frac{1}{|X \times Y|} \sum_{(x,y)\in X\times Y} (f(x, y) - z) \right|.$$

This expresses the discrepancy as the maximum difference between the average values of $f(x, y)$ and the constant function z.

2. **Upper Bound of $C^Z(f)$**: The correlation $C^Z(f)$ is defined as the inner product between f and the constant function $Z(x, y) = z$. Thus, we have:

$$C^Z(f) = \frac{1}{|X \times Y|} \sum_{(x,y)\in X\times Y} (f(x, y) - z).$$

This represents the average difference between $f(x, y)$ and the constant function Z over all $(x, y) \in X \times Y$.

3. **Relationship Between $D(f)$ and $C^Z(f)$**: From the definitions above, we observe that:

$$D(f) = \max_{z \in \{0,1\}} \left| \frac{1}{|X \times Y|} \sum_{(x,y) \in X \times Y} (f(x, y) - z) \right| = |C^Z(f)| + 1.$$

This inequality follows from the fact that the discrepancy is bounded by the correlation with the constant function Z plus 1.

Therefore, we conclude that $D(f) \leq C^Z(f) + 1$, as required. □

D.1.3 Claude

Proof. Let $f : X \times Y \to \{0, 1\}$ be arbitrary and let $z \in \{0, 1\}$. We need to show that $D(f) \leq C^Z(f)+1$.

Let P be an optimal protocol for f in the zero-error communication model with cost z, so that $C^Z(f) = \text{cost}_z(P)$. We construct a deterministic protocol P' for f as follows.

On input $(x, y) \in X \times Y$, the protocol P' simulates P on (x, y). During this simulation, whenever P would send a bit b with cost z, the protocol P' sends the bit b. Whenever P would send a bit b with cost $1 - z$ (the free bit), the protocol P' sends the bit b as well.

Since P correctly computes $f(x, y)$ for all inputs (x, y), the protocol P' also correctly computes $f(x, y)$ for all inputs.

Now we analyze the communication cost of P'. The total number of bits sent by P' on input (x, y) equals the total number of bits sent by P on input (x, y), which is at most $\text{cost}_z(P) + 1$. This is because in the zero-error model with cost z, the protocol P sends at most $\text{cost}_z(P)$ bits of type z and at most one bit of type $1 - z$ (since typically in such models, we count only the expensive bits and allow at most one free bit).

More precisely, in the standard definition of $C^Z(f)$, we have that $C^Z(f)$ counts the number of bits equal to z sent in the worst case, and protocols are allowed to send at most one additional bit equal to $1 - z$ without cost. Therefore, the total number of bits communicated is at most $C^Z(f) + 1$.

Since P' is a valid deterministic protocol for f that communicates at most $C^Z(f) + 1$ bits, and $D(f)$ is defined as the minimum communication cost over all deterministic protocols for f, we have

$$D(f) \leq \text{cost}(P') \leq C^Z(f) + 1.$$

This completes the proof. □

D.1.4 Gemini

Proof. Let $f : X \times Y \to \{0, 1\}$ be a boolean function and let $z \in \{0, 1\}$. We define $C^Z(f)$ as the one-sided deterministic communication complexity of f with respect to the value z. Specifically, $C^Z(f)$ is the minimum cost of a deterministic protocol $\mathcal{P}$ that satisfies the following conditions for any input $(x, y) \in X \times Y$:

1. If $f(x, y) = z$, then $\mathcal{P}(x, y)$ outputs z.
2. If $f(x, y) \neq z$, then $\mathcal{P}(x, y)$ outputs "reject" (denoted as $\perp$).

To prove that $D(f) \leq C^Z(f) + 1$, we construct a deterministic protocol $\mathcal{P}'$ for f using the optimal protocol $\mathcal{P}$ associated with $C^Z(f)$.

Protocol $\mathcal{P}'$:

1. Alice and Bob simulate the protocol $\mathcal{P}$ on input (x, y).
2. Let v be the output of $\mathcal{P}$.
3. If $v = z$, the parties output z.
4. If $v = \perp$, the parties output $1 - z$.

Correctness: Since f is a boolean function with range $\{0,1\}$, for any (x,y):

- Case 1: $f(x,y)=z$. By the definition of $\mathcal{P}$, the output is $v=z$. $\mathcal{P}'$ outputs z, which is correct.
- Case 2: $f(x,y)\neq z$. Since the range is binary, this implies $f(x,y)=1-z$. By the definition of $\mathcal{P}$, the output is $v=\perp$. $\mathcal{P}'$ outputs $1-z$, which is correct.

Thus, $\mathcal{P}'$ computes f correctly on all inputs.

Complexity: The cost of determining v is exactly the cost of $\mathcal{P}$, which is $C^Z(f)$.If the protocol $\mathcal{P}$ ensures that both parties know the output v at a leaf, then the cost of $\mathcal{P}'$ is exactly $C^Z(f)$.If the termination of $\mathcal{P}$ implies that only one party knows the result (e.g., in a one-way communication setting or if the leaf is determined by one party's local view), one additional bit of communication may be required to ensure both parties agree on the final output value.In the worst-case scenario, the communication complexity of $\mathcal{P}'$ is bounded by:

$$D(f)\leq \text{cost}(\mathcal{P})+1=C^Z(f)+1$$

Therefore, $D(f)\leq C^Z(f)+1$ holds for all $z\in\{0,1\}$. □

D.1.5 Grok

Proof. Let $f: X\times Y\to\{0,1\}$ be arbitrary, and fix $z\in\{0,1\}$. Without loss of generality, assume $z=1$; the case $z=0$ follows by applying the argument to the negation $\bar{f}$, noting that $D(f)=D(\bar{f})$ and $C^0(f)=C^1(\bar{f})$.

Let $k=C^1(f)$. By definition, there exists a collection of k rectangles $\{R_j=A_j\times B_j\mid 1\leq j\leq k\}$ such that each R_j is 1-monochromatic (i.e., $f(x,y)=1$ for all $(x,y)\in A_j\times B_j$) and every (x,y) with $f(x,y)=1$ belongs to at least one R_j.

Consider the following deterministic protocol:

Alice computes the vector $v\in\{0,1\}^k$ defined by $v_j=1$ if $x\in A_j$ and $v_j=0$ otherwise.

Alice sends v to Bob, using k bits of communication.

Bob computes the vector $w\in\{0,1\}^k$ defined by $w_j=1$ if $y\in B_j$ and $w_j=0$ otherwise.

Bob checks whether $v\wedge w\neq\mathbf{0}^k$ (the all-zero vector), i.e., whether there exists some j such that $v_j=w_j=1$.

If yes, Bob outputs 1; otherwise, Bob outputs 0.

(If the model requires both parties to know the output, Bob sends the 1-bit result to Alice.)

The total communication is at most $k+1$ bits.

To verify correctness:

Suppose $f(x,y)=1$. Then $(x,y)\in R_j$ for some j (by the covering property), so $x\in A_j$ and $y\in B_j$. Thus, $v_j=1$ and $w_j=1$, so $v\wedge w\neq\mathbf{0}^k$ and the output is 1.

Suppose $f(x,y)=0$. Then $(x,y)\notin R_j$ for all j (since each R_j is 1-monochromatic). Thus, there is no j such that $x\in A_j$ and $y\in B_j$, so there is no j with $v_j=w_j=1$. Hence, $v\wedge w=\mathbf{0}^k$ and the output is 0.

The protocol correctly computes f. Therefore,

$$D(f)\leq k+1=C^1(f)+1.$$

The argument for $z=0$ is symmetric.

Thus, $D(f)\leq C^z(f)+1$ for all $z\in\{0,1\}$. □

D.2 Problem 2

D.2.1 Problem Statement

Theorem D.2. *Show that both $N^0(GT)$ and $N^1(GT)$ are at least n.*

D.2.2 GPT

Proof. Let G be a graph with n vertices. We aim to show that both $N^0(GT)$ and $N^1(GT)$ are at least n.

First, recall the definitions of $N^0(GT)$ and $N^1(GT)$:

$$N^0(GT) = \{v \in V(G) : \deg(v) \geq 0\}$$

$$N^1(GT) = \{v \in V(G) : \deg(v) \geq 1\}$$

We will prove that both $N^0(GT)$ and $N^1(GT)$ contain at least n vertices.

Step 1: Show that $N^0(GT)$ contains at least n vertices

By definition, $N^0(GT)$ consists of all vertices in the graph G whose degree is greater than or equal to 0. Since the degree of every vertex in a graph is always non-negative, it follows that every vertex in G belongs to $N^0(GT)$. Thus, we have:

$$|N^0(GT)| = |V(G)| = n$$

Therefore, $N^0(GT)$ contains at least n vertices.

Step 2: Show that $N^1(GT)$ contains at least n vertices

Next, we consider $N^1(GT)$, which consists of all vertices in G whose degree is greater than or equal to 1. We will prove that $N^1(GT)$ contains at least n vertices by showing that there are no isolated vertices in G.

Suppose, for the sake of contradiction, that G contains an isolated vertex, say v. This means that $\deg(v) = 0$, so v would not belong to $N^1(GT)$. However, since every vertex in G has degree at least 1, it follows that no vertex in G can be isolated. Hence, there are no isolated vertices in G.

Since every vertex in G has degree at least 1, it follows that every vertex belongs to $N^1(GT)$. Thus, we have:

$$|N^1(GT)| = |V(G)| = n$$

Therefore, $N^1(GT)$ contains at least n vertices.

Since both $N^0(GT)$ and $N^1(GT)$ contain at least n vertices, the theorem is proven. □

D.2.3 Claude

Proof. We will show that $N^0(GT) \geq n$ and $N^1(GT) \geq n$.

First, we establish that $N^0(GT) \geq n$. Recall that $N^0(GT)$ denotes the number of vertices in the graph GT. By the construction of GT, each of the n original vertices in G corresponds to at least one vertex in GT. Specifically, the vertex set of GT contains all vertices from the original graph G, which has n vertices. Therefore, $|V(GT)| \geq n$, which implies $N^0(GT) \geq n$.

Next, we prove that $N^1(GT) \geq n$. Recall that $N^1(GT)$ denotes the number of edges in the graph GT. Consider the spanning tree T of G. Since T is a spanning tree of a graph with n vertices, T contains exactly $n - 1$ edges.

In the construction of GT, we include all edges from the spanning tree T. Additionally, for each edge $e \in E(G) \setminus E(T)$ (that is, each edge in G that is not in T), we add at least one corresponding edge or structure to GT. Since G is connected and has n vertices, and T is a spanning tree with $n - 1$ edges, there exists at least one edge in $E(G) \setminus E(T)$ (otherwise G would be a tree itself).

Therefore, GT contains at least the $n - 1$ edges from T plus at least one additional edge from the non-tree edges, giving us at least n edges in total. Hence, $N^1(GT) \geq n$.

Thus, we have shown that both $N^0(GT) \geq n$ and $N^1(GT) \geq n$. □

D.2.4 Gemini

Proof. Let $GT_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be the Greater-Than function, defined as $GT_n(x, y) = 1$ if the integer represented by x is strictly greater than the integer represented by y, and 0 otherwise. We treat x and y as integers in the range $[0, 2^n - 1]$.The values $N^0(GT)$ and $N^1(GT)$ denote the nondeterministic communication complexities of computing $GT_n(x, y) = 0$ and $GT_n(x, y) = 1$, respectively. These are

bounded from below by the logarithm of the size of a fooling set for the respective output values.Formally, if S is a fooling set for value $v \in \{0,1\}$, then $N^v(GT) \geq \lceil \log_2 |S| \rceil$.

Part 1: Lower bound for $N^0(GT)$ (the case $x \leq y$) Let S_0 be the set of input pairs defined by the diagonal elements:

$$S_0 = \{(x,x) \mid x \in \{0,1,\ldots,2^n-1\}\}$$

For every $(x,x) \in S_0$, we have $x \leq x$, so $GT_n(x,x) = 0$. Thus, S_0 is a subset of the 0-inputs.Consider any two distinct pairs (x,x) and (y,y) from S_0. Without loss of generality, assume $x > y$ (as integers).We examine the cross-pair (x,y). Since $x > y$, we have $GT_n(x,y) = 1$.Since $GT_n(x,y) \neq 0$, the rectangle covering (x,x) and (y,y) cannot be monochromatic with value 0.Therefore, S_0 is a fooling set for the 0-inputs of size $|S_0| = 2^n$.The nondeterministic complexity is bounded by:

$$N^0(GT) \geq \log_2(2^n) = n$$

Part 2: Lower bound for $N^1(GT)$ (the case $x > y$) Let S_1 be the set of input pairs defined as:

$$S_1 = \{(x+1,x) \mid x \in \{0,1,\ldots,2^n-2\}\}$$

For every pair $(x+1,x) \in S_1$, we have $x+1 > x$, so $GT_n(x+1,x) = 1$. Thus, S_1 is a subset of the 1-inputs.Consider any two distinct pairs $(x+1,x)$ and $(y+1,y)$ from S_1. Assume without loss of generality that $x < y$.Since x and y are integers, $x < y$ implies $x+1 \leq y$.We examine the cross-pair $(x+1,y)$. Since $x+1 \leq y$, we have $GT_n(x+1,y) = 0$.Since $GT_n(x+1,y) \neq 1$, the rectangle covering $(x+1,x)$ and $(y+1,y)$ cannot be monochromatic with value 1.Therefore, S_1 is a fooling set for the 1-inputs of size $|S_1| = 2^n - 1$.The nondeterministic complexity is bounded by:

$$N^1(GT) \geq \lceil \log_2(2^n-1) \rceil$$

For $n \geq 2$, $2^{n-1} < 2^n - 1 < 2^n$, which implies $\lceil \log_2(2^n-1) \rceil = n$.Thus, for $n \geq 2$, $N^1(GT) \geq n$.Combining both parts, we have shown that $N^0(GT) \geq n$ and $N^1(GT) \geq n$. □

D.2.5 Grok

Proof. The inputs x, y are integers with $0 \leq x, y < 2^n$. The function satisfies $\mathrm{GT}(x,y) = 1$ if and only if $x > y$, and $\mathrm{GT}(x,y) = 0$ if and only if $x \leq y$. Here $N^z(\mathrm{GT})$ denotes the nondeterministic communication complexity for verifying that $\mathrm{GT}(x,y) = z$ (the minimum length of certificates in a nondeterministic protocol accepting precisely when $\mathrm{GT}(x,y) = z$).

A nondeterministic protocol verifying $\mathrm{GT}(x,y) = z$ corresponds to a cover of the z-entries by 2^c z-monochromatic rectangles, where $c = N^z(\mathrm{GT})$ (the prover sends the index of a rectangle covering the input, and the parties verify membership in their side).

Thus, lower bounds on the z-cover number $C^z(\mathrm{GT})$ (minimum number of z-monochromatic rectangles covering the z-entries) imply $N^z(\mathrm{GT}) \geq \lceil \log_2 C^z(\mathrm{GT}) \rceil$.

First, consider $z = 1$. A 1-monochromatic rectangle is a set $A \times B$ such that $a > b$ for all $a \in A$, $b \in B$, or equivalently $\min A > \max B$.

Let $P = \{(k+1,k) \mid 0 \leq k \leq 2^n - 2\}$. Then $|P| = 2^n - 1$ and each element of P is a 1-entry.

Suppose a 1-monochromatic rectangle $A \times B$ contains two distinct elements $(k+1,k), (l+1,l) \in P$ with $k < l$ (without loss of generality). Then $\{k+1, l+1\} \subseteq A$ and $\{k,l\} \subseteq B$, so $\min A \leq k+1$ and $\max B \geq l$. Since $l \geq k+1$, it follows that $\max B \geq l \geq k+1 \geq \min A$, contradicting $\min A > \max B$.

Thus, every 1-monochromatic rectangle contains at most one element of P. Since the 1-entries must be covered and $P \subseteq$ the 1-entries, $C^1(\mathrm{GT}) \geq 2^n - 1$.

Since $2^{n-1} \leq 2^n - 1 < 2^n$ for $n \geq 1$ and $2^n - 1 > 2^{n-1}$, we have $n - 1 < \log_2(2^n - 1) < n$, so $\lceil \log_2(2^n - 1) \rceil = n$. Therefore, $N^1(\mathrm{GT}) \geq n$.

Now consider $z = 0$. A 0-monochromatic rectangle is a set $A \times B$ such that $a \leq b$ for all $a \in A$, $b \in B$, or equivalently $\max A \leq \min B$.

Let $D = \{(k,k) \mid 0 \leq k \leq 2^n - 1\}$. Then $|D| = 2^n$ and each element of D is a 0-entry.

Suppose a 0-monochromatic rectangle $A \times B$ contains two distinct elements $(k,k),(l,l) \in D$ with $k < l$ (without loss of generality). Then $\{k,l\} \subseteq A$ and $\{k,l\} \subseteq B$, so $\max A \geq l$ and $\min B \leq k$. Since $l > k$, it follows that $\max A \geq l > k \geq \min B$, contradicting $\max A \leq \min B$.

Thus, every 0-monochromatic rectangle contains at most one element of D. Since $D \subseteq$ the 0-entries, $C^0(\mathrm{GT}) \geq 2^n$.

Therefore, $N^0(\mathrm{GT}) \geq \lceil \log_2 2^n \rceil = n$.

Hence, both $N^0(\mathrm{GT})$ and $N^1(\mathrm{GT})$ are at least n. □

D.3 Problem 3

D.3.1 Problem Statement

Theorem D.3. *Improve the constant in the above upper bound to $D(f) \leq 3\log_2 C^P(f)$. Hint: find a node v that splits the tree into three parts, each of them with at most $t/2$ nodes.*

D.3.2 GPT

Proof. Let T be a binary tree with t nodes, and let f be a function defined on the nodes of T. We aim to improve the constant in the upper bound for $D(f)$, where $D(f)$ is defined as the depth of the tree T with respect to the function f, and show that $D(f) \leq 3\log_2 C^P(f)$.

To begin, consider the function f and the associated binary tree T. Let the value $C^P(f)$ denote the complexity of f with respect to the tree T. The task is to find a node v that splits the tree T into three parts such that each part contains at most $\frac{t}{2}$ nodes. We will do this by analyzing the tree structure and the branching properties of T.

We first note that any binary tree with t nodes has a depth of at least $\log_2 t$ and at most t. To improve the bound, we need to partition the tree in such a way that we can exploit the relationship between the depth of the tree and the function f. By the properties of binary trees, there always exists a node v in the tree T that splits the tree into three subtrees. Let the subtrees rooted at v be denoted as T_1, T_2, and T_3.

By construction, each of these subtrees contains at most $\frac{t}{2}$ nodes. The depth of the tree T is determined by the maximum depth of the subtrees T_1, T_2, and T_3. Let the depths of these subtrees be denoted as $D(T_1)$, $D(T_2)$, and $D(T_3)$. Then, the depth of the entire tree T is given by:

$$D(T) = 1 + \max(D(T_1), D(T_2), D(T_3)).$$

Since each subtree contains at most $\frac{t}{2}$ nodes, the depth of each subtree is at most $\log_2\left(\frac{t}{2}\right) = \log_2 t - 1$. Therefore, we have:

$$D(T) \leq 1 + \max(\log_2 t - 1, \log_2 t - 1, \log_2 t - 1) = \log_2 t.$$

Thus, the depth of the tree T is at most $\log_2 t$. We now use the fact that the complexity $C^P(f)$ satisfies $C^P(f) \geq \log_2 t$. From this, it follows that:

$$D(f) \leq 3\log_2 C^P(f).$$

Hence, we have improved the constant in the upper bound, and the desired inequality is established. □

D.3.3 Claude

Proof. Let f be a Boolean function and let T be an optimal decision tree computing f with $t = C^P(f)$ leaves. We prove that $D(f) \leq 3\log_2 C^P(f)$ by constructing a decision tree of depth at most $3\log_2 t$.

We proceed by induction on t. For the base case, if $t = 1$, then f is constant and $D(f) = 0 \leq 3\log_2 1 = 0$.

For the inductive step, assume $t \geq 2$ and the result holds for all functions with certificate complexity strictly less than t. Consider the optimal decision tree T with t leaves. We construct a balanced decision tree as follows.

Let v be a node in T such that each of the three subtrees formed by removing v and its two children contains at most $t/2$ leaves. To show such a node exists, we use the following procedure: Start at the root. If the current node v satisfies the property, we are done. Otherwise, at least one of the three parts (left

subtree, right subtree, or the complement) contains more than $t/2$ leaves. Since the total is t leaves and two parts have at most $t/2$ leaves each, exactly one subtree of v contains more than $t/2$ leaves. Move to the root of that subtree and repeat. This process must terminate since we move down the tree and eventually reach a node where both subtrees have at most $t/2$ leaves.

More formally, let v be a node where both children have subtrees with at most $t/2$ leaves. Then the left subtree of v has at most $t/2$ leaves, the right subtree of v has at most $t/2$ leaves, and the remaining part of T (excluding the subtree rooted at v) has at most $t/2$ leaves.

Query the variable at node v first. This splits the problem into at most three subproblems:

- The left subtree of v with at most $t/2$ leaves
- The right subtree of v with at most $t/2$ leaves
- The path from root to v may create additional branches, but these are absorbed into the recursion

By the inductive hypothesis, each subproblem can be solved with depth at most $3\log_2(t/2) = 3\log_2 t - 3$.

Therefore, the total depth is at most:

$$1 + 3\log_2(t/2) = 1 + 3\log_2 t - 3 = 3\log_2 t - 2 \leq 3\log_2 t$$

for $t \geq 2$.

Thus, $D(f) \leq 3\log_2 C^P(f)$. □

D.3.4 Gemini

Proof. Let T be a deterministic protocol tree for the function f with size $S = C^P(f)$, where S denotes the number of leaves in T. We prove the bound $D(f) \leq 3\log_2 S$ by induction on S.

Base Case: If $S = 1$, the tree consists of a single leaf labeled with the output. No communication is required, so $D(f) = 0 \leq 3\log_2 1$.

Inductive Step: Assume the hypothesis holds for all trees with fewer than S leaves. Let T be a protocol tree with S leaves. For any node $u \in T$, let $L(u)$ denote the number of leaves in the subtree rooted at u. We identify a "centroid" node v in T using the following procedure: Start at the root. If the current node has a child with more than $S/2$ leaves, descend to that child. Repeat this process until a node v is reached such that every child of v has at most $S/2$ leaves. Since we only descend to a child if its weight exceeds $S/2$, the node v satisfies:

$$L(v) > S/2 \quad \text{and} \quad L(v_{\text{left}}) \leq S/2, \quad L(v_{\text{right}}) \leq S/2,$$

where v_{left} and v_{right} are the children of v. Note that since $L(v) > S/2$ and $S \geq 2$, v cannot be a leaf.

The node v corresponds to a rectangle $R_v = X_v \times Y_v$ such that the protocol reaches v if and only if the input $(x, y) \in R_v$. The protocol proceeds as follows:

1. Alice sends a bit $a = 1$ if $x \in X_v$, and 0 otherwise.
2. Bob sends a bit $b = 1$ if $y \in Y_v$, and 0 otherwise.

This determines whether the computation path passes through v.

Case 1: $a = 1$ and $b = 1$. The input lies in R_v, so the execution has reached node v. Since v is an internal node, it is owned by either Alice or Bob. The owner computes the next step (Left or Right) based on their input and sends 1 bit to indicate the direction. The protocol then proceeds in either the left subtree T_{left} or the right subtree T_{right}. The number of leaves in the subsequent tree is at most $S/2$. By the induction hypothesis, the remaining complexity is bounded by $3\log_2(S/2)$. The total cost is:

$$2 + 1 + 3\log_2(S/2) = 3 + 3(\log_2 S - 1) = 3\log_2 S.$$

Case 2: $a = 0$ or $b = 0$. The input is not in R_v, meaning the execution path diverges from the path to v at some ancestor of v. This effectively prunes the subtree rooted at v from T. We can view this as

executing a protocol on a modified tree T_{out} where the subtree at v is replaced by a single leaf (representing the unreachable state "path went to v"). The number of leaves in T_{out} is $S - L(v) + 1$. Since $L(v) > S/2$, we have:

$$L(T_{\text{out}}) < S - S/2 + 1 = S/2 + 1.$$

For integer sizes, this implies $L(T_{\text{out}}) \approx S/2$. Strictly, if S is even, $L(v) \geq S/2+1 \implies L(T_{\text{out}}) \leq S/2$. If S is odd, $L(T_{\text{out}}) \leq \lceil S/2 \rceil$. The cost in this step is 2 bits. By the induction hypothesis on T_{out}, the remaining complexity is bounded by $3\log_2(S/2)$. Thus:

$$2 + 3\log_2(S/2) = 2 + 3\log_2 S - 3 = 3\log_2 S - 1 \leq 3\log_2 S.$$

In both cases, the total communication complexity is at most $3\log_2 S$. □

D.3.5 Grok

Proof. We prove by induction on $t = C^P(f)$ that $D(f) \leq 3\log_2 t$.

The base cases for small t (e.g., $t \leq 8$) hold trivially, as $D(f)$ is at most the number of relevant variables, which is small enough to satisfy the bound.

Assume the statement holds for all functions with $C^P(g) < t$, where $t > 8$.

Consider a minimal structure (tree) associated with the measure $C^P(f) = t$.

There exists a node v such that removing v splits the tree into three parts, each with at most $t/2$ nodes.

Each part corresponds to a subproblem with measure at most $t/2$.

We construct a decision tree for f with root querying the variable associated with v, and recursively attaching the decision trees for the three subproblems.

The depth satisfies the recurrence $D(f) \leq 3 + \max\{D(g_1), D(g_2), D(g_3)\}$, where each $C^P(g_i) \leq t/2$.

By the induction hypothesis,

$$D(g_i) \leq 3\log_2(t/2) = 3\log_2 t - 3.$$

Thus,

$$D(f) \leq 3 + (3\log_2 t - 3) = 3\log_2 t.$$

This completes the induction. □

D.4 Problem 4

D.4.1 Problem Statement

Problem D.4 (Open Problem 2.10)**.** *Is $D(f) = O(\log C^D(f))$?*

D.4.2 GPT

Proof. We are tasked with proving whether $D(f) = O(\log C^D(f))$. To do this, we will carefully examine the definitions and properties of the involved terms.

Let us assume that $D(f)$ and $C^D(f)$ are well-defined functions. We begin by recalling the definition of Big-O notation. We say that $D(f) = O(\log C^D(f))$ if there exist constants $C > 0$ and n_0 such that for all $n \geq n_0$:

$$|D(f)| \leq C\log(C^D(f))$$

First, observe that $C^D(f)$ is an expression dependent on the function f, and the logarithmic term is applied to it. For large values of $C^D(f)$, $\log(C^D(f))$ behaves asymptotically as a standard logarithmic growth. Thus, we aim to find suitable constants C and n_0 such that the inequality holds.

Now, let us rewrite the inequality as follows:

$$D(f) \leq C\log(C^D(f))$$

For this inequality to hold, we examine the asymptotic growth of both sides. The function $D(f)$ must grow slower than or at most logarithmically with respect to $C^D(f)$, as logarithms typically grow slower than polynomial functions.

Since the logarithmic function grows slower than any polynomial function, and assuming that $D(f)$ itself is bounded by some polynomial growth of $C^D(f)$, we conclude that there exists a constant $C > 0$ such that $D(f) = O(\log C^D(f))$. This is because, in the worst case, the growth of $D(f)$ is comparable to or slower than the logarithmic growth of $C^D(f)$.

Thus, we have shown that $D(f) = O(\log C^D(f))$ holds for suitable constants, completing the proof. □

D.4.3 Claude

Proof. This is an open problem, and we present the current state of knowledge regarding the relationship between deterministic query complexity $D(f)$ and deterministic communication complexity $C^D(f)$.

We first establish what is known. For any Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, we have the trivial upper bound $D(f) \leq n$ since any deterministic decision tree can query at most all n bits.

By the standard simulation argument, we know that $C^D(f) \geq \log(L(f))$ where $L(f)$ denotes the leaf complexity of the optimal decision tree for f. Furthermore, we have $D(f) = \log_2(L(f))$ when the decision tree is complete and balanced.

The best known upper bound is $D(f) = O(C^D(f))$, which follows from the fact that Alice and Bob can simulate a decision tree protocol: at each node, the player holding the relevant variable sends one bit indicating the outcome, requiring $D(f)$ bits of communication total. This gives $C^D(f) \geq D(f)$.

For the converse direction, we can construct a decision tree from a communication protocol. Given a protocol using $C^D(f)$ bits, we can build a decision tree where each communication bit corresponds to a sequence of queries. However, this naive construction yields $D(f) = O(2^{C^D(f)})$ in the worst case, as there are $2^{C^D(f)}$ possible transcripts.

The question asks whether $D(f) = O(\log C^D(f))$, which would imply $C^D(f) = 2^{\Omega(D(f))}$. This is equivalent to asking whether communication complexity must be exponential in decision tree depth for some functions.

Current separation results show that there exist functions where $C^D(f) = \Omega(D(f))$, but no function is known where $C^D(f) = 2^{\Omega(D(f))}$. The strongest known lower bound is $C^D(f) = \Omega(D(f) \cdot \log D(f))$ for certain functions.

Therefore, the question of whether $D(f) = O(\log C^D(f))$ holds for all Boolean functions remains open. A negative answer would require exhibiting a function f with $D(f) = \omega(\log C^D(f))$, while a positive answer would require proving $C^D(f) = 2^{\Omega(D(f))}$ for all functions, which contradicts known upper bounds.

Thus, this problem remains unresolved in computational complexity theory. □

D.4.4 Gemini

Proof. The statement asks whether the deterministic communication complexity $D(f)$ is bounded linearly by the logarithm of the partition number $C^D(f)$ (often denoted $\chi(f)$), i.e., whether $D(f) = O(\log C^D(f))$. This is a well-known open problem in communication complexity, closely related to the P^{cc} vs. NP^{cc} question. While the linear bound is open (and known to be false in strictly stronger forms such as $P^{cc} = NP^{cc}$ being refuted by counterexamples showing polynomial separations), we can prove the following quadratic upper bound:

$$D(f) = O((\log C^D(f))^2).$$

Let $\mathcal{R} = \{R_1, \ldots, R_k\}$ be a minimal partition of the communication matrix M_f into f-monochromatic rectangles, so $k = C^D(f)$. The nondeterministic communication complexity $N(f)$ is defined as $\lceil \log_2 \mathrm{cov}^1(f) \rceil$, where $\mathrm{cov}^1(f)$ is the minimum number of 1-monochromatic rectangles covering the 1-inputs. Similarly, $N(\neg f) = \lceil \log_2 \mathrm{cov}^0(f) \rceil$. Since a partition covers both 1-inputs and 0-inputs using disjoint sets, we have:

$$\mathrm{cov}^1(f) \leq C^D(f) \quad \text{and} \quad \mathrm{cov}^0(f) \leq C^D(f).$$

Consequently,

$$N(f) \leq \lceil \log_2 C^D(f) \rceil \quad \text{and} \quad N(\neg f) \leq \lceil \log_2 C^D(f) \rceil.$$

A classic result by Aho, Ullman, and Yannakakis establishes that deterministic complexity is bounded by the product of the nondeterministic complexities of f and $\neg f$:

$$D(f) \leq O(N(f) \cdot N(\neg f)).$$

Substituting the bounds derived from the partition number:

$$D(f) \leq O(\log C^D(f) \cdot \log C^D(f)) = O((\log C^D(f))^2).$$

Thus, while it is proven that $D(f)$ is at most quadratic in $\log C^D(f)$, the question of whether this can be improved to a linear bound $O(\log C^D(f))$ remains the subject of Open Problem 2.10. □

D.4.5 Grok

Proof. The statement asks whether $D(f) = O(\log C^D(f))$ for every Boolean function f, where $D(f)$ denotes the deterministic decision tree depth and $C^D(f)$ is a complexity measure (likely a variant of certificate complexity or a related quantity distinct from the partition-type measure $C^P(f)$ in prior results).

This question is presented as Open Problem 2.10 in the relevant context, indicating that it remains unresolved.

No general upper bound of the form $D(f) \leq O(\log C^D(f))$ is known, nor is there a known counterexample disproving it for the specific definition of $C^D(f)$.

The stronger logarithmic bound would require a splitting argument that balances into two parts (yielding a recurrence $D(f) \leq 1 + \max D(g_i)$ and hence $D(f) \leq O(\log t)$), in contrast to the three-part splitting that yields the factor of 3 in the bound $D(f) \leq 3 \log_2 C^P(f)$.

Whether such a two-way balanced separator always exists for the minimal structure defining $C^D(f)$ is open.

Thus, the problem remains open. □

D.5 Problem 5

D.5.1 Problem Statement

Theorem D.5. *Recall the definition of geometric rectangles given in Exercise 1.18. Let f be any function for which $X \times Y$ can be covered by t f-monochromatic geometric rectangles (possibly with overlaps). Prove that $D(f) = O(\log t)$.*

D.5.2 GPT

Proof. Let f be a function for which the product space $X \times Y$ can be covered by t f-monochromatic geometric rectangles, possibly with overlaps. We are tasked with proving that $D(f) = O(\log t)$, where $D(f)$ represents the complexity of the function f.

Recall that $D(f)$ is defined as the smallest number of distinct pieces needed to cover the domain of f such that each piece is f-monochromatic. That is, for any covering of $X \times Y$ by f-monochromatic sets, the complexity $D(f)$ counts the minimal number of such sets required to cover the domain.

Since $X \times Y$ is covered by t f-monochromatic geometric rectangles, we know that there exists a collection $\mathcal{R}$ of at most t such rectangles, where each rectangle in $\mathcal{R}$ is f-monochromatic. Denote these rectangles as $R_1, R_2, \ldots, R_t$. These rectangles form a covering of $X \times Y$ by f-monochromatic sets.

Each rectangle R_i corresponds to a specific f-monochromatic set, and the collection $\mathcal{R}$ covers the entire domain $X \times Y$. Therefore, the number of distinct f-monochromatic sets needed to cover $X \times Y$ is at most t.

Next, consider the complexity of the function f. The key observation is that f-monochromatic sets can be represented as unions of geometric rectangles, which are defined by their boundaries. A crucial property of these geometric rectangles is that their complexity is bounded by the logarithm of the number of such rectangles in a covering. Specifically, the number of distinct pieces (rectangles) required to cover $X \times Y$ grows logarithmically with the number of rectangles.

Thus, the complexity $D(f)$ is proportional to the logarithm of the number of rectangles required to cover the domain, which is at most t. Therefore, we conclude that

$$D(f) = O(\log t)$$

This completes the proof. □

D.5.3 Claude

Proof. We prove that if $X \times Y$ can be covered by t f-monochromatic geometric rectangles, then $D(f) = O(\log t)$.

Recall that a geometric rectangle is a set of the form $A \times B$ where $A \subseteq X$ and $B \subseteq Y$, and such a rectangle is f-monochromatic if f is constant on $A \times B$. The deterministic communication complexity $D(f)$ is the minimum depth of any deterministic protocol tree computing f.

Let $\mathcal{R} = \{R_1, R_2, \ldots, R_t\}$ be a collection of f-monochromatic geometric rectangles covering $X \times Y$, where each $R_i = A_i \times B_i$ for some $A_i \subseteq X$ and $B_i \subseteq Y$.

We construct a deterministic protocol for computing f as follows. The protocol proceeds in rounds, where in each round we eliminate at least half of the remaining rectangles from consideration.

At the start, all t rectangles are active. Given input $(x, y) \in X \times Y$, we know that $(x, y) \in R_i$ for some $i \in \{1, 2, \ldots, t\}$.

In each round, Alice and Bob perform the following procedure:

Round procedure: Let S denote the current set of active rectangles. Alice examines which rectangles $R_i = A_i \times B_i$ in S contain her input x, i.e., for which $x \in A_i$. Bob examines which rectangles in S contain his input y, i.e., for which $y \in B_i$.

Alice and Bob use a binary search strategy. They partition S into two sets S_1 and S_2 of roughly equal size (with $|S_1| \leq \lceil |S|/2 \rceil$ and $|S_2| \leq \lceil |S|/2 \rceil$). Alice sends one bit indicating whether there exists a rectangle in S_1 containing x (i.e., whether $x \in \bigcup_{R_i \in S_1} A_i$). Bob then sends one bit indicating whether there exists a rectangle in S_1 containing y (i.e., whether $y \in \bigcup_{R_i \in S_1} B_i$).

If both bits are 1, then (x, y) must be in some rectangle in S_1, so they set $S := S_1$. Otherwise, (x, y) must be in some rectangle in S_2, so they set $S := S_2$.

After each round, the number of active rectangles is reduced by at least a factor of 2. Starting with t rectangles, after k rounds, at most $t/2^k$ rectangles remain active.

The protocol terminates when exactly one rectangle remains active, say $R_j = A_j \times B_j$. Since $(x, y) \in R_j$ and R_j is f-monochromatic with value v_j, both parties output $f(x, y) = v_j$.

The number of rounds required is at most $\lceil \log_2 t \rceil$, since we need $t/2^k \leq 1$, which gives $k \geq \log_2 t$.

Each round uses 2 bits of communication, so the total communication is at most $2\lceil \log_2 t \rceil = O(\log t)$.

Therefore, $D(f) \leq 2\lceil \log_2 t \rceil = O(\log t)$. □

D.5.4 Gemini

Proof. Let X and Y be totally ordered sets. By the definition of geometric rectangles, any geometric rectangle $R \subseteq X \times Y$ can be expressed as $R = I \times J$, where $I \subseteq X$ and $J \subseteq Y$ are intervals. Let $\mathcal{R} = \{R_1, R_2, \ldots, R_t\}$ be the given cover of $X \times Y$ by t f-monochromatic geometric rectangles. For each $k \in \{1, \ldots, t\}$, we write $R_k = I_k \times J_k$.

We define a grid on $X \times Y$ induced by the endpoints of these intervals. Let E_X be the set of all endpoints of the intervals $I_1, \ldots, I_t$. Specifically, if X is a discrete domain $\{1, \ldots, n\}$, for an interval $I_k = [a_k, b_k]$, we consider the boundary points necessary to partition the space. Generally, the set of endpoints of t intervals partitions the set X into at most $2t + 1$ disjoint elementary intervals (or atomic sets). Let $\mathcal{I} = \{A_1, A_2, \ldots, A_m\}$ be this partition of X, where $m \leq 2t + 1$. This partition satisfies the property that for any elementary interval $A_\alpha \in \mathcal{I}$ and any rectangle interval I_k, either $A_\alpha \subseteq I_k$ or $A_\alpha \cap I_k = \emptyset$.

Similarly, let E_Y be the set of endpoints of the intervals $J_1, \ldots, J_t$. These endpoints partition Y into a collection of disjoint elementary intervals $\mathcal{J} = \{B_1, B_2, \ldots, B_n\}$, where $n \leq 2t + 1$. This partition satisfies the property that for any $B_\beta \in \mathcal{J}$ and any J_k, either $B_\beta \subseteq J_k$ or $B_\beta \cap J_k = \emptyset$.

Consider the product partition $\mathcal{G} = \{A_\alpha \times B_\beta \mid 1 \leq \alpha \leq m, 1 \leq \beta \leq n\}$. The elements of $\mathcal{G}$ form a grid of cells covering $X \times Y$. We claim that f is constant on each cell $C_{\alpha,\beta} = A_\alpha \times B_\beta$.

Let (x, y) be an arbitrary point in a cell $C_{\alpha,\beta}$. Since $\mathcal{R}$ covers $X \times Y$, there exists an index $k \in \{1, \ldots, t\}$ such that $(x, y) \in R_k$. By definition, $R_k = I_k \times J_k$, so $x \in I_k$ and $y \in J_k$. Since $x \in A_\alpha$ and $A_\alpha \cap I_k \neq \emptyset$, the refinement property implies $A_\alpha \subseteq I_k$. Similarly, since $y \in B_\beta$ and $B_\beta \cap J_k \neq \emptyset$, we have $B_\beta \subseteq J_k$. Therefore, $C_{\alpha,\beta} = A_\alpha \times B_\beta \subseteq I_k \times J_k = R_k$.

Because R_k is f-monochromatic, f takes a constant value $v_k \in \{0, 1\}$ on all points in R_k. Since $C_{\alpha,\beta} \subseteq R_k$, f must be constant on the entire cell $C_{\alpha,\beta}$ (specifically, $f(x', y') = v_k$ for all $(x', y') \in C_{\alpha,\beta}$). Note that while a cell might be covered by multiple overlapping rectangles, the function f is well-defined, so all rectangles covering the cell must share the same monochromatic value.

We can now describe a deterministic communication protocol for f: Alice receives input x and Bob receives input y. Alice identifies the index α such that $x \in A_\alpha$. Since there are $m \leq 2t + 1$ such intervals, she can encode α using $\lceil \log(2t + 1) \rceil$ bits. Alice sends this encoding to Bob. Bob identifies the index β such that $y \in B_\beta$. Since there are $n \leq 2t + 1$ such intervals, he can encode β using $\lceil \log(2t + 1) \rceil$ bits. Bob sends this encoding to Alice. Both parties now know the indices α and β, which uniquely identify the cell $C_{\alpha,\beta}$. Since f is constant on this cell, the value of $f(x, y)$ is determined solely by the pair (α, β). Both parties output this value.

The total number of bits communicated is:

$$\lceil \log(2t + 1) \rceil + \lceil \log(2t + 1) \rceil = O(\log t).$$

Thus, the deterministic communication complexity satisfies $D(f) = O(\log t)$. □

D.5.5 Grok

Proof. Let $R_1, \ldots, R_t$ be the t f-monochromatic geometric rectangles that cover $X \times Y$. For each $k = 1, \ldots, t$, write $R_k = I_k \times J_k$, where $I_k \subseteq X$ and $J_k \subseteq Y$ are intervals, and f is constant on R_k with value $c_k \in \{0, 1\}$.

Consider the intervals $I_1, \ldots, I_t$ on the ordered set X. The at most $2t$ endpoints of these intervals divide X into at most $2t + 1$ contiguous blocks $B_1, \ldots, B_s$ (where $s \leq 2t + 1$), ordered naturally, such that for every x in the same block B_j, the set $S_j = \{k \mid x \in I_k\}$ is identical.

Both parties precompute this partitioning of X into the s blocks and the corresponding sets S_j for $j = 1, \ldots, s$.

Alice, holding input $x \in X$, determines the unique j such that $x \in B_j$ and sends the index j to Bob, using at most $\lceil \log_2 s \rceil \leq \lceil \log_2(2t + 1) \rceil$ bits.

Bob, upon receiving j, retrieves the set $S_j = \{k \mid x \in I_k\}$. Since $X \times Y = \bigcup_{k=1}^{t} R_k$, there exists at least one $k \in S_j$ such that $y \in J_k$. Bob finds any such k and sets $v = c_k$. As $f(x, y) = c_k$ for every k with $(x, y) \in R_k$, we have $v = f(x, y)$.

Bob sends the bit v to Alice.

The total number of bits communicated is at most $\lceil \log_2(2t + 1) \rceil + 1 = O(\log_2 t)$. Therefore, $D(f) = O(\log_2 t)$. □

D.6 Problem 6

D.6.1 Problem Statement

Theorem D.6 (Exercise 2.18*)**.** *Most functions $f : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}$ satisfy $N^f(r) = \Omega(n)$, and yet the size of the largest fooling set for f is $O(n)$. (An explicit example of a function with these properties is given in Example 4.16.)*

D.6.2 GPT

Proof. Let $f : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}$ be a boolean function. The statement asserts that for most functions f, we have $N^f(r) = \Omega(n)$, where $N^f(r)$ is the sensitivity of f at input r, and that the size of the largest fooling set for f is $O(n)$.

To begin the proof, recall the definitions:

1. The sensitivity $N^f(r)$ of the function f at an input $r \in \{0, 1\}^n \times \{0, 1\}^n$ is the number of coordinates in which changing the value of f at r causes the output of f to flip. Formally, for a given $r = (x, y)$ where

$x, y \in \{0,1\}^n$, we define $N^f(r)$ as:

$$N^f(r) = |\{(r') \in \{0,1\}^n \times \{0,1\}^n : f(r) \neq f(r')\}|.$$

2. A fooling set for f is a set of input pairs $\{(x_1, y_1), \ldots, (x_k, y_k)\}$ such that for every distinct pair (x_i, y_i) and (x_j, y_j) in the set, we have:

$$f(x_i, y_i) \neq f(x_j, y_j).$$

The size of the largest fooling set is denoted by $|FS(f)|$.

Sensitivity Lower Bound

Let us first demonstrate that for most functions f, the sensitivity $N^f(r)$ is large. Consider the function f defined by the majority vote of its inputs. For such functions, f depends on many bits, and hence for most choices of r, the sensitivity is large. In particular, for any fixed $r = (x, y)$, the number of inputs that cause the output to flip is proportional to n, and thus we have $N^f(r) = \Omega(n)$ for most functions. This can be verified by examining the structure of boolean functions that are sensitive to changes in their inputs, such as parity or majority functions, which exhibit sensitivity proportional to the number of variables.

Fooling Set Upper Bound

Next, we show that the size of the largest fooling set is small, specifically $O(n)$. Consider a function f with the property that its value is determined by a small number of bits or by a simple combinatorial structure, such as a threshold function or a function defined by a linear code. For these functions, the number of input pairs that can be used to form a fooling set is limited by the number of possible distinct outputs that f can produce.

By considering the structure of f, we observe that for most boolean functions, the largest fooling set cannot be too large because the function typically satisfies some form of locality or has a small number of "critical" bits that determine its output. This implies that the size of the largest fooling set for f is $O(n)$. In particular, if f is defined by a sparse linear combination of its inputs or by a threshold, the number of input pairs that can fool f is bounded by $O(n)$, since each distinct pair must correspond to a distinct output, and the number of such pairs is limited by the number of possible outputs.

Conclusion

Thus, for most functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, we have $N^f(r) = \Omega(n)$, indicating that the sensitivity is large, while simultaneously, the size of the largest fooling set for f is $O(n)$, indicating that the function does not have a large number of distinct input pairs that can fool it.

Therefore, the theorem is proved. □

D.6.3 Claude

Proof. We prove this theorem by a probabilistic argument showing that a random function has the desired properties with high probability.

Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be chosen uniformly at random. We need to show that with high probability, $N^f(r) = \Omega(n)$ while the largest fooling set has size $O(n)$.

Part 1: Lower bound on $N^f(r)$.

For a random function f, consider the nondeterministic communication complexity $N^f(r)$. By definition, $N^f(r)$ is the minimum number of rectangles needed to cover all 1-entries of f.

The total number of inputs is 2^{2n}. For a random function, approximately half the inputs map to 1, so there are roughly 2^{2n-1} ones to cover. Each rectangle in a nondeterministic protocol corresponds to a pair (S, T) where $S \subseteq \{0,1\}^n$ and $T \subseteq \{0,1\}^n$.

Consider rectangles of a fixed size. If we use r rectangles to cover the 1-entries, and each rectangle has area at most A, then we can cover at most rA entries. For the covering to be possible, we need $rA \geq 2^{2n-1}$.

By a counting argument, if $r = o(n)$, then with high probability over the choice of f, no collection of r rectangles can cover all 1-entries. Specifically, the probability that a fixed collection of r rectangles covers all 1-entries is at most $2^{-\Omega(2^{2n}/r)}$, which is exponentially small. Taking a union bound over all

possible collections of r rectangles (of which there are at most $2^{O(rn)}$), we find that for $r = o(n)$, no such covering exists with high probability.

Therefore, $N^f(r) = \Omega(n)$ with high probability.

Part 2: Upper bound on fooling set size.

A fooling set S for f is a set of pairs (x_i, y_i) such that $f(x_i, y_i) = 1$ for all i, and for all $i \neq j$, either $f(x_i, y_j) = 0$ or $f(x_j, y_i) = 0$.

For a random function f, consider any potential fooling set of size s. Fix pairs $(x_1, y_1), \ldots, (x_s, y_s)$. The probability that this forms a fooling set is the probability that:

(i) $f(x_i, y_i) = 1$ for all $i \in [s]$, which occurs with probability 2^{-s}.

(ii) For all $i \neq j$, at least one of $f(x_i, y_j) = 0$ or $f(x_j, y_i) = 0$ holds.

For condition (ii), for each pair (i, j) with $i \neq j$, the probability that both $f(x_i, y_j) = 1$ and $f(x_j, y_i) = 1$ is at most $1/4$ (assuming these are distinct from the diagonal entries).

The number of potential fooling sets of size s is at most 2^{2ns} (choosing s pairs from $\{0,1\}^n \times \{0,1\}^n$).

By the union bound, the probability that there exists a fooling set of size s is at most:

$$2^{2ns} \cdot 2^{-s} \cdot (3/4)^{-\binom{s}{2}} \leq 2^{2ns - s + O(s^2)}.$$

For $s = \omega(n)$, this probability tends to 0 as $n \to \infty$. Therefore, with high probability, the largest fooling set has size $O(n)$.

Combining both parts, we conclude that most functions satisfy $N^f(r) = \Omega(n)$ while having largest fooling set of size $O(n)$. □

D.6.4 Gemini

Proof. We consider a random Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ chosen uniformly from the set of all such functions. The total number of Boolean functions is $2^{2^{2n}}$.

Part 1: Lower bound on $N(f)$. Recall that $N(f) = \lceil \log_2 \mathrm{cov}^1(f) \rceil$, where $\mathrm{cov}^1(f)$ is the minimum number of monochromatic rectangles needed to cover the 1-inputs of f. A function has $N(f) \leq r$ if and only if its support $f^{-1}(1)$ can be written as the union of at most 2^r rectangles. The number of possible combinatorial rectangles $A \times B$ in $\{0,1\}^n \times \{0,1\}^n$ is $(2^{2^n})^2 = 2^{2^{n+1}}$. Let $\mathcal{N}_r$ be the number of functions with $N(f) \leq r$. This is bounded by the number of ways to choose 2^r rectangles from the set of all possible rectangles:

$$\mathcal{N}_r \leq \binom{2^{2^{n+1}}}{2^r} \leq \left(2^{2^{n+1}}\right)^{2^r} = 2^{2^r \cdot 2^{n+1}} = 2^{2^{n+r+1}}.$$

For a random function, we require $\mathcal{N}_r \ll 2^{2^{2n}}$. Consider $r = n - 2$. The number of such functions is at most $2^{2^{2n-1}}$, which is the square root of the total number of functions. Thus, the fraction of functions with $N(f) \leq n - 2$ is at most $2^{2^{2n-1}} / 2^{2^{2n}} = 2^{-2^{2n-1}}$, which approaches 0 rapidly. Therefore, for most functions, $N(f) > n - 2$, implying $N(f) = \Omega(n)$.

Part 2: Upper bound on the fooling set size. Let k be an integer. A set of input pairs $S = \{(x_1, y_1), \ldots, (x_k, y_k)\}$ is a fooling set for f if: 1. $f(x_i, y_i) = 1$ for all $i \in \{1, \ldots, k\}$. 2. For all $i \neq j$, either $f(x_i, y_j) = 0$ or $f(x_j, y_i) = 0$.

We bound the probability that a random function f has a fooling set of size k. Fix a sequence of k distinct pairs $P = ((x_1, y_1), \ldots, (x_k, y_k))$. For P to be a fooling set, the values of f on the diagonal must be 1 (probability $(1/2)^k$). For the off-diagonal condition, consider the pairs (x_i, y_j) and (x_j, y_i) for $i < j$. There are $\binom{k}{2}$ such pairs of indices. Assuming the inputs allow for independent checks (which holds loosely or can be bounded by the worst case), the probability that the condition "$f(x_i, y_j) = 0 \vee f(x_j, y_i) = 0$" holds for a specific pair $\{i, j\}$ is $1 - (1/2)^2 = 3/4$. Thus, the probability that a fixed set P is a fooling set is at most:

$$(1/2)^k \left(\frac{3}{4}\right)^{\binom{k}{2}}.$$

The number of choices for the set P is at most $\binom{2^{2n}}{k} \leq (2^{2n})^k = 2^{2nk}$. By the union bound, the probability that there exists a fooling set of size k is bounded by:

$$\Pr[\exists \text{ fooling set of size } k] \leq 2^{2nk} \cdot 2^{-k} \cdot \left(\frac{3}{4}\right)^{k(k-1)/2}.$$

Taking the logarithm base 2 of the bound:

$$\log_2(\text{Bound}) \approx 2nk - k - \frac{k^2}{2}\log_2(4/3).$$

Let $c = \frac{1}{2}\log_2(4/3) \approx 0.207$. The exponent is roughly $2nk - ck^2$. For the probability to vanish, we need the quadratic term to dominate, i.e., $ck^2 > 2nk \implies k > \frac{2n}{c} \approx 10n$. Thus, if we choose $k = Cn$ for a sufficiently large constant C, the probability that a fooling set of size k exists approaches 0. Consequently, for most functions, the size of the largest fooling set is $O(n)$.

Combining both parts, most functions satisfy $N(f) = \Omega(n)$ while having a largest fooling set of size $O(n)$. □

D.6.5 Grok

Proof. Consider a uniformly random function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, where each of the 2^{2n} entries is independently 0 or 1 with probability $1/2$.

A fooling set of size s is a set of pairs $\{(x_i, y_i)\}_{i=1}^s$ with distinct x_i, distinct y_i, $f(x_i, y_i) = 1$ for all i, and $f(x_i, y_j) = 0$ for all $i \neq j$.

The number of candidate fooling sets of size s is at most $\binom{2^n}{s}^2 s!$.

For any fixed candidate, the probability that it is a fooling set is 2^{-s^2}, as s entries must be 1 and $s(s-1)$ entries must be 0.

By the union bound, the probability that a fooling set of size s exists is at most

$$\binom{2^n}{s}^2 s! \cdot 2^{-s^2}.$$

Using $\binom{2^n}{s} \leq (e \cdot 2^n/s)^s$ and $s! \leq (es)^s$,

$$\binom{2^n}{s}^2 s! \leq (e \cdot 2^n/s)^{2s}(es)^s = 2^{2sn} e^{3s} s^{-s}.$$

Thus, the probability is at most

$$2^{2sn - s^2 + 3s\log_2 e - s\log_2 s}.$$

Let $s = 100n$. Then $2sn - s^2 = 200n^2 - 10000n^2 = -9800n^2$, and the remaining terms are $O(n \log n)$. Hence, the probability is $2^{-\Omega(n^2)}$.

With probability $1 - 2^{-\Omega(n^2)}$, no fooling set of size $100n$ exists. Since the constant 100 is arbitrary (any sufficiently large constant works), the largest fooling set has size $O(n)$ with high probability.

Now consider the nondeterministic communication complexity $N^f(r)$, defined as the minimum c such that there exists a nondeterministic protocol of cost c verifying $f(x, y) = 1$ (i.e., the 1-entries are exactly the union of $2^{O(c)}$ 1-monochromatic rectangles).

The number of functions with $N^f(r) \leq c$ is at most the number of ways to choose 2^c rectangles, which is at most $(2^{2^{n+1}})^{2^c} = 2^{2^{n+1} \cdot 2^c}$.

Let $c = n/100$. Then $2^{n+1} \cdot 2^{n/100} = 2^{O(n)}$, so there are at most $2^{2^{O(n)}}$ such functions.

The total number of functions is $2^{2^{2n}}$. Since $2^{O(n)} = o(2^{2n})$, the number of functions with $N^f(r) \leq n/100$ is $o(2^{2^{2n}}/2^{2^{2n}})$, i.e., the probability is $o(1)$.

Thus, with probability $1 - o(1)$, $N^f(r) = \Omega(n)$.

Therefore, with high probability, $N^f(r) = \Omega(n)$ and the largest fooling set has size $O(n)$. □

D.7 Problem 7

D.7.1 Problem Statement

Problem D.7. *Does* $D(f) = (\log \operatorname{rank}(f))^{O(1)}$ *for all* $f : X \times Y \to \{0,1\}$*?*

D.7.2 GPT

Proof. We are tasked with determining whether the statement $D(f) = (\log \operatorname{rank}(f))^{O(1)}$ holds for all functions $f : X \times Y \to \{0,1\}$, where $D(f)$ denotes the depth of the decision tree for f and $\operatorname{rank}(f)$ represents the rank of the matrix f.

First, recall that the decision tree complexity $D(f)$ of a Boolean function f is the minimum depth of a decision tree that computes f. This depth is related to the complexity of computing the function based on queries to the input variables.

The rank of the matrix f, $\operatorname{rank}(f)$, is the rank of the matrix whose entries are given by $f(x, y)$ for all $x \in X$ and $y \in Y$. The matrix M_f has rows indexed by elements of X and columns indexed by elements of Y, where the entry in row x and column y is given by $f(x, y)$. The rank of this matrix reflects the linear dependence between rows or columns of the matrix.

We now seek to establish the relation between $D(f)$ and $\operatorname{rank}(f)$.

First, we observe that if f has low rank, the entries of the matrix M_f are highly constrained. Specifically, if $\operatorname{rank}(f) = r$, then the matrix M_f can be expressed as a sum of at most r outer products of vectors. Thus, the matrix has a low degree of freedom, and we expect that any function f with small rank can be computed by a decision tree with relatively small depth.

Conversely, if f has high rank, the matrix M_f has a large degree of freedom, suggesting that f may require a deeper decision tree to compute. In this case, the rank of the matrix imposes fewer constraints on the structure of the function, and a decision tree may need to explore more possibilities before determining the value of $f(x, y)$ for a given input.

Next, we use the fact that decision tree depth $D(f)$ is closely related to the rank of the matrix M_f. It is known that for any Boolean function f, the decision tree depth is upper bounded by a polynomial in the rank of M_f. Specifically, it holds that:

$$D(f) \leq (\operatorname{rank}(f))^{O(1)}$$

This upper bound arises from the fact that the rank of f governs the complexity of the function's representation, and the decision tree depth is constrained by the linear structure of the matrix M_f. The rank of f directly impacts the number of distinct decision paths required to compute f, thus bounding the depth of the decision tree.

Therefore, we conclude that for all functions $f : X \times Y \to \{0,1\}$, the decision tree depth satisfies the desired inequality:

$$D(f) = (\log \operatorname{rank}(f))^{O(1)}$$

Thus, the statement is true. □

D.7.3 Claude

Proof. We prove that the statement $D(f) = (\log \operatorname{rank}(f))^{O(1)}$ does not hold for all Boolean functions $f : X \times Y \to \{0,1\}$ by constructing a counterexample.

Let n be a positive integer and consider the Inner Product function $\mathrm{IP}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ defined by

$$\mathrm{IP}_n(x, y) = \sum_{i=1}^{n} x_i y_i \pmod 2.$$

We establish lower and upper bounds for the deterministic communication complexity $D(\mathrm{IP}_n)$ and the rank $\operatorname{rank}(\mathrm{IP}_n)$ of the communication matrix.

Lower bound for $D(\mathrm{IP}_n)$: By a standard argument using the fooling set method or by appealing to known results in communication complexity, we have $D(\mathrm{IP}_n) = n$. To see this, observe that any

deterministic protocol must distinguish between all 2^n possible values of x when y is the standard basis vector e_i, which requires at least n bits of communication.

Upper bound for $\mathrm{rank}(\mathbf{IP}_n)$: The communication matrix M of IP_n is a $2^n \times 2^n$ matrix where $M_{x,y} = \mathrm{IP}_n(x, y)$. This matrix can be written as

$$M = \bigotimes_{i=1}^{n} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \bigotimes_{i=1}^{n} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \pmod 2.$$

Over $\mathbb{F}_2$, the rank of M is at most 2^n trivially, but more importantly, it is known that $\mathrm{rank}_{\mathbb{F}_2}(\mathrm{IP}_n) = 2^n$. However, when considering rank over $\mathbb{R}$, we have $\mathrm{rank}_{\mathbb{R}}(M) \leq 2^n$ as well.

More precisely, by the relationship between communication complexity and rank, we know that $\log \mathrm{rank}(f) \leq D(f)$ always holds. For the inner product function, it is known that $\mathrm{rank}_{\mathbb{R}}(\mathrm{IP}_n) = 2^n$, which gives $\log \mathrm{rank}(\mathrm{IP}_n) = n$.

Comparison: We have established that

$$D(\mathrm{IP}_n) = n \quad \text{and} \quad \log \mathrm{rank}(\mathrm{IP}_n) = n.$$

Therefore, for the inner product function, $D(\mathrm{IP}_n) = \Theta(n)$ and $\log \mathrm{rank}(\mathrm{IP}_n) = \Theta(n)$, which means

$$D(\mathrm{IP}_n) = \Theta(\log \mathrm{rank}(\mathrm{IP}_n)).$$

However, this does not provide a counterexample. We need a different function.

Consider instead the Disjointness function $\mathrm{DISJ}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ where $\mathrm{DISJ}_n(x, y) = 1$ if and only if $x_i \cdot y_i = 0$ for all $i \in [n]$.

It is known that $D(\mathrm{DISJ}_n) = \Omega(n)$ by results of Kalyanasundaram and Schnitger, while $\mathrm{rank}(\mathrm{DISJ}_n) = 2^{\Theta(n)}$, giving $\log \mathrm{rank}(\mathrm{DISJ}_n) = \Theta(n)$.

Thus, for disjointness, we have $D(\mathrm{DISJ}_n) = \Theta(n) = \Theta(\log \mathrm{rank}(\mathrm{DISJ}_n))$, which again shows the relationship is tight up to constants.

Therefore, the answer is: Yes, it is conjectured and widely believed (though not fully proven for all functions) that $D(f) = (\log \mathrm{rank}(f))^{O(1)}$ holds for all Boolean functions f, with the exponent typically being 1 or 2 depending on the specific bounds used. □

D.7.4 Gemini

Proof. The statement questions the validity of the Log-Rank Conjecture, which asserts that the deterministic communication complexity $D(f)$ is polylogarithmic in the rank of the communication matrix M_f over the reals. While the conjecture remains an open problem, we provide the formal proof for the established lower bound, $D(f) \geq \log_2 \mathrm{rank}(f)$, and discuss the limitations of current upper bounds.

Let M_f be the communication matrix of f, where $(M_f)_{x,y} = f(x, y)$. Let $\mathcal{P}$ be a deterministic protocol for f with communication complexity $c = D(f)$. The protocol $\mathcal{P}$ can be represented by a binary tree of depth c. The leaves of this tree form a partition of the input space $X \times Y$ into combinatorial rectangles $\mathcal{R} = \{R_1, \ldots, R_L\}$, where $L \leq 2^c$ is the number of leaves.

Since the protocol computes f deterministically, each rectangle R_i is f-monochromatic. Let $\mathcal{R}_1 = \{R \in \mathcal{R} \mid \forall (x, y) \in R, f(x, y) = 1\}$ be the set of 1-monochromatic rectangles. The communication matrix M_f can be expressed as the sum of the characteristic matrices of these rectangles:

$$M_f = \sum_{R \in \mathcal{R}_1} M_R,$$

where M_R is the matrix defined by $(M_R)_{x,y} = 1$ if $(x, y) \in R$ and 0 otherwise.

Since each $R \in \mathcal{R}_1$ is a rectangle, it can be written as $A \times B$ for some $A \subseteq X, B \subseteq Y$. The matrix M_R can be written as the outer product of the characteristic vectors $\mathbf{1}_A$ and $\mathbf{1}_B$, i.e., $M_R = \mathbf{1}_A \mathbf{1}_B^T$. Consequently, $\mathrm{rank}(M_R) = 1$ for every non-empty rectangle R.

Using the subadditivity of the rank function, we have:

$$\operatorname{rank}(M_f) = \operatorname{rank}\left(\sum_{R\in\mathcal{R}_1} M_R\right) \le \sum_{R\in\mathcal{R}_1} \operatorname{rank}(M_R) = |\mathcal{R}_1|.$$

Since $\mathcal{R}_1$ is a subset of the leaves of the protocol tree, $|\mathcal{R}_1| \le L \le 2^{D(f)}$. Therefore,

$$\operatorname{rank}(f) \le 2^{D(f)} \implies D(f) \ge \log_2 \operatorname{rank}(f).$$

This establishes that $D(f) = \Omega(\log \operatorname{rank}(f))$.

Regarding the upper bound, it is known that $D(f) \le \operatorname{rank}(f) + 1$. The Log-Rank Conjecture asks whether this can be tightened to $(\log \operatorname{rank}(f))^{O(1)}$. The current best known upper bound, due to Lovett (2014), is $D(f) = O(\sqrt{\operatorname{rank}(f)} \log \operatorname{rank}(f))$, which leaves the question of a polylogarithmic bound open. □

D.7.5 Grok

Proof. The question asks whether there exists a universal constant c such that $D(f) \le (\log \operatorname{rank}(f))^c$ for every communication matrix $f : X \times Y \to \{0, 1\}$, where $\operatorname{rank}(f)$ denotes the rank of the communication matrix of f over $\mathbb{R}$.

It is well known that $D(f) \ge \log_2 \operatorname{rank}(f)$. Indeed, any deterministic protocol of cost c partitions the matrix into at most 2^c monochromatic rectangles, so the matrix is the disjoint union of at most 2^c rank-1 matrices, implying $\operatorname{rank}(f) \le 2^c$.

The converse direction, an upper bound of the form $D(f) \le (\log \operatorname{rank}(f))^{O(1)}$, is a longstanding open question known as the log-rank conjecture.

No proof of such a polynomial upper bound is known. The best unconditional upper bounds on $D(f)$ in terms of $\operatorname{rank}(f)$ are significantly weaker (exponential or quasipolynomial in $\sqrt{\log \operatorname{rank}(f)}$).

Moreover, explicit functions are known for which $D(f) \ge (\log \operatorname{rank}(f))^{\omega(1)}$, showing that if the conjecture holds, the exponent must be superconstant.

No counterexample disproving the existence of some fixed polynomial bound is known.

Therefore, the question remains open. □

D.8 Problem 8

D.8.1 Problem Statement

Theorem D.8 (Exercise 2.21)**.** *Let f be a Boolean function. Then:*

(a) $D(f) = O(N^0(f) \log \textit{rank}(f))$

(b) $D(f) = O(N^1(f) \log \textit{rank}(f))$

where $D(f)$ denotes the deterministic communication complexity of f, $N^0(f)$ denotes the 0-monochromatic rectangle cover number, $N^1(f)$ denotes the 1-monochromatic rectangle cover number, and rank(f) denotes the rank of the communication matrix of f.

D.8.2 GPT

Proof. We are tasked with proving the following two parts of the theorem:

$$D(f) = O(N^0(f) \log \operatorname{rank}(f)) \quad \text{and} \quad D(f) = O(N^1(f) \log \operatorname{rank}(f))$$

where $D(f)$ denotes the deterministic communication complexity of a Boolean function f, $N^0(f)$ and $N^1(f)$ denote the 0-monochromatic and 1-monochromatic rectangle cover numbers, respectively, and $\operatorname{rank}(f)$ denotes the rank of the communication matrix of f.

Part (a): $D(f) = O(N^0(f) \log \operatorname{rank}(f))$

To begin, recall that the deterministic communication complexity $D(f)$ is defined as the minimum number of bits exchanged between two parties, Alice and Bob, when they are each given an input, x and y, respectively, and must compute $f(x, y)$ deterministically.

The communication matrix of f is a $2^m \times 2^n$ matrix, where m and n are the bit lengths of the inputs of Alice and Bob, respectively. The rank of this matrix, $\text{rank}(f)$, represents the minimum number of rows that span the entire matrix.

Now, consider the 0-monochromatic rectangle cover number $N^0(f)$, which is defined as the smallest number of 0-monochromatic rectangles required to cover the communication matrix. A 0-monochromatic rectangle is a submatrix where all entries are 0. The key observation is that these 0-monochromatic rectangles represent possible ways to restrict the communication to certain areas where the function f consistently outputs 0.

By a result from communication complexity theory, the deterministic communication complexity $D(f)$ is bounded above by the product of the 0-monochromatic rectangle cover number and the logarithm of the rank of the communication matrix. Specifically, we have the inequality:

$$D(f) \leq N^0(f) \log \text{rank}(f)$$

Thus, we conclude that:

$$D(f) = O(N^0(f) \log \text{rank}(f))$$

as desired.

Part (b): $D(f) = O(N^1(f) \log \text{rank}(f))$ Similarly, the 1-monochromatic rectangle cover number $N^1(f)$ is defined as the smallest number of 1-monochromatic rectangles required to cover the communication matrix. A 1-monochromatic rectangle is a submatrix where all entries are 1.

Analogous to the previous case, we can use the fact that the deterministic communication complexity is also bounded by the 1-monochromatic rectangle cover number and the logarithm of the rank of the communication matrix. Specifically, we have:

$$D(f) \leq N^1(f) \log \text{rank}(f)$$

Therefore, we conclude:

$$D(f) = O(N^1(f) \log \text{rank}(f))$$

as required.

Thus, both parts of the theorem are proven. □

D.8.3 Claude

Proof. We prove both parts of the theorem.

Part (a): We show that $D(f) = O(N^0(f) \log \text{rank}(f))$.

Let $\mathcal{R} = \{R_1, R_2, \ldots, R_k\}$ be a minimal cover of $f^{-1}(0)$ by 0-monochromatic rectangles, where $k = N^0(f)$. By the definition of deterministic communication complexity, we can construct a protocol that computes f as follows.

For each rectangle R_i in the cover, we have $R_i = A_i \times B_i$ for some $A_i \subseteq \{0,1\}^n$ and $B_i \subseteq \{0,1\}^m$. Since R_i is a rectangle, it corresponds to a rank-1 submatrix of the communication matrix M_f. By the properties of matrix rank, each such rectangle R_i has rank at most 1.

The key observation is that any monochromatic rectangle R in the communication matrix satisfies $\text{rank}(R) \leq \text{rank}(M_f)$. Since we have $k = N^0(f)$ rectangles covering $f^{-1}(0)$, and the communication matrix can be partitioned into at most $N^0(f) + N^1(f)$ monochromatic rectangles, we can use a protocol tree where each internal node corresponds to distinguishing between rectangles.

To distinguish between k rectangles, we need at most $\lceil \log_2 k \rceil$ bits of communication. However, we must also account for the complexity of identifying which rectangle contains the input pair (x, y). By the log-rank conjecture framework, for each rectangle R_i, the parties can determine membership in R_i using $O(\log \text{rank}(M_f))$ bits of communication.

The protocol proceeds as follows: Alice and Bob use a binary search over the rectangles in $\mathcal{R}$. At each step, they determine whether (x, y) lies in a particular subset of rectangles. This requires $O(\log \text{rank}(M_f))$ bits per query. After $O(\log N^0(f))$ queries, they identify the unique rectangle (if any) containing (x, y) where $f(x, y) = 0$. If no such rectangle exists, then $f(x, y) = 1$.

Therefore, the total communication complexity is:

$$D(f) = O(\log N^0(f) \cdot \log \text{rank}(M_f)) = O(N^0(f) \log \text{rank}(f))$$

where we use the fact that $\log N^0(f) \leq N^0(f)$ for all positive integers $N^0(f)$.

Part (b): We show that $D(f) = O(N^1(f) \log \text{rank}(f))$.

The proof is symmetric to part (a). Let $\mathcal{S} = \{S_1, S_2, \ldots, S_\ell\}$ be a minimal cover of $f^{-1}(1)$ by 1-monochromatic rectangles, where $\ell = N^1(f)$.

By the same reasoning as in part (a), Alice and Bob can construct a protocol that searches over the rectangles in $\mathcal{S}$. They determine whether (x, y) lies in any rectangle $S_j \in \mathcal{S}$ where f evaluates to 1. Each membership query requires $O(\log \text{rank}(M_f))$ bits of communication.

Using binary search over the $\ell = N^1(f)$ rectangles, the parties need $O(\log N^1(f))$ queries, each costing $O(\log \text{rank}(M_f))$ bits. If (x, y) lies in some rectangle S_j, then $f(x, y) = 1$; otherwise, $f(x, y) = 0$.

Therefore, the total communication complexity is:

$$D(f) = O(\log N^1(f) \cdot \log \text{rank}(M_f)) = O(N^1(f) \log \text{rank}(f))$$

where again we use $\log N^1(f) \leq N^1(f)$.

This completes the proof of both parts. □

D.8.4 Gemini

Proof. We prove part (b): $D(f) = O(N^1(f) \log \text{rank}(f))$. Part (a) follows by symmetry, as $D(f) = D(\neg f)$, $N^0(f) = N^1(\neg f)$, and $\text{rank}(\neg f) \leq \text{rank}(f) + 1$. Note that here $N^1(f)$ denotes the nondeterministic communication complexity, i.e., $N^1(f) = \lceil \log_2 \text{cov}^1(f) \rceil$.

Let $r = \text{rank}(f)$. We construct a deterministic protocol for f. The strategy relies on the fact that if a function f has a small 1-cover, we can decompose the problem based on the rank. Let $\mathcal{C} = \{R_1, \ldots, R_{2^{N^1(f)}}\}$ be a minimal 1-cover of f. The function f computes to 1 if and only if the input (x, y) belongs to at least one rectangle in $\mathcal{C}$.

We proceed by induction on $N^1(f)$. Base case: If $N^1(f) = 0$, then f is covered by a single rectangle $R = A \times B$. Thus $f(x, y) = \mathbb{I}_A(x) \wedge \mathbb{I}_B(y)$. Alice sends 1 bit indicating if $x \in A$, and Bob sends 1 bit indicating if $y \in B$. The total communication is 2, and the bound holds since $\log r \geq 0$.

Inductive step: Assume the bound holds for any function with nondeterministic complexity less than k. Let f be a function with $N^1(f) = k$. We split the cover $\mathcal{C}$ into two disjoint sets $\mathcal{C}_1$ and $\mathcal{C}_2$ of equal size 2^{k-1}. Let f_1 be the function supported by $\bigcup_{R \in \mathcal{C}_1} R$ and f_2 be the function supported by $\bigcup_{R \in \mathcal{C}_2} R$. Then $f = f_1 \vee f_2$. Note that $\text{rank}(f_1) \leq \text{rank}(f)$ is not necessarily true, but we can use a linear algebraic property. Specifically, the rows of the communication matrix M_f span a space V of dimension r. Alice and Bob can determine whether $f(x, y) = 1$ by computing $f_1(x, y)$ and $f_2(x, y)$. However, simply computing both would yield a recurrence $D(k) \leq 2D(k - 1)$, which is exponential.

Instead, we use a protocol that checks consistency with the row space. Consider the 2^{k-1} rectangles in $\mathcal{C}_1$. Let S_1 be the support of f_1. Alice and Bob perform the following check: Does there exist a row in the matrix M_{f_1} that is consistent with Alice's input x and Bob's input y? This is equivalent to checking if the projection of the inputs onto the row space of f (or a related subspace) is non-zero? Actually, the standard proof utilizes the following result by Nisan and Wigderson (1995): There exists a protocol that reduces the size of the cover candidate set by half, using $O(\log \text{rank}(f))$ bits of communication. Let S be the current set of candidate rectangles from the cover $\mathcal{C}$. Initially $|S| = 2^{N^1(f)}$. In each phase, Alice and Bob communicate $O(\log r)$ bits to determine a subset $S' \subset S$ with $|S'| \leq |S|/2$ such that if $f(x, y) = 1$, then the covering rectangle must be in S'. This repeats $N^1(f)$ times. If S becomes empty and no 1 has been confirmed, output 0. The cost of such a step is bounded by $\log \text{rank}(f)$ using the fact that for any set of rectangles, the question "is (x, y) covered by the first half?" can be approximated or checked via rank properties (specifically, verifying linear dependencies among rows restricted to the cover).

Thus, the total communication complexity is:

$$D(f) \leq N^1(f) \cdot O(\log \text{rank}(f)) = O(N^1(f) \log \text{rank}(f)).$$

Combining this with part (a) via symmetry completes the proof. □

D.8.5 Grok

Proof. We prove part (a); the proof of part (b) is analogous by replacing 0 with 1 and $N^0(f)$ with $N^1(f)$.

Let $m = N^0(f)$. Let $\{R_j = A_j \times B_j \mid j = 1, \ldots, m\}$ be a minimum cover of the 0-entries of the communication matrix by 0-monochromatic rectangles, that is, $f(x, y) = 0$ implies $(x, y) \in R_j$ for some j, and $f(x, y) = 0$ for all $(x, y) \in R_j$.

Consider the following deterministic protocol:

Alice, on input x, constructs the vector $s \in \{0, 1\}^m$ where the j-th component $s_j = 1$ if $x \in A_j$ and $s_j = 0$ otherwise. Alice sends s to Bob (m bits of communication).

Bob, on input y, constructs the vector $t \in \{0, 1\}^m$ where $t_j = 1$ if $y \in B_j$ and $t_j = 0$ otherwise. Bob computes the inner product $s \cdot t = \sum_{j=1}^{m} s_j t_j$. If $s \cdot t > 0$, output 0; otherwise, output 1.

The communication complexity of this protocol is m bits.

For correctness: Suppose $f(x, y) = 0$. Then there exists some j such that $(x, y) \in R_j$, so $x \in A_j$ and $y \in B_j$, hence $s_j = t_j = 1$ and $s \cdot t \geq 1 > 0$, so the output is 0.

Suppose $f(x, y) = 1$. Then $(x, y) \notin R_j$ for all j, so there is no j with $x \in A_j$ and $y \in B_j$ simultaneously. Thus $s_j t_j = 0$ for all j, so $s \cdot t = 0$, and the output is 1.

Since each R_j is 0-monochromatic, no 1-entry is contained in any R_j, so the cases are exhaustive and the protocol correctly computes f.

Thus $D(f) \leq m = N^0(f)$. Since $\log \operatorname{rank}(f) \geq 0$, it follows that $D(f) = O(N^0(f) \log \operatorname{rank}(f))$.

The same protocol, with 0 and 1 interchanged, shows $D(f) \leq N^1(f) = O(N^1(f) \log \operatorname{rank}(f))$.

□

D.9 Problem 9

D.9.1 Problem Statement

Theorem D.9 (Exercise 2.22)**.** *Let* $\operatorname{Mono}(f)$ *denote the fraction of* M_f*'s entries covered by the largest monochromatic rectangle of* f*. Show that if for every function* f*,* $\log(1/\operatorname{Mono}(f)) = (\log \operatorname{rank}(f))^{O(1)}$ *when Open Problem 2.20 gets a positive answer. Hint: Look at the proof of Theorem 2.11 (Combinatorial Version).*

D.9.2 GPT

Proof. Let f be a function, and let M_f denote the matrix corresponding to f. Denote by $\operatorname{Mono}(f)$ the fraction of M_f's entries covered by the largest monochromatic rectangle of f. We are tasked with showing that if Open Problem 2.20 has a positive answer, then the following inequality holds for all functions f:

$$\log\left(\frac{1}{\operatorname{Mono}(f)}\right) = (\log \operatorname{rank}(f))^{O(1)}.$$

Step 1: Recall the result from Theorem 2.11 (Combinatorial Version)

We begin by referring to Theorem 2.11 (Combinatorial Version), which provides a bound on the rank of a matrix in terms of the size of the largest monochromatic rectangle in the matrix. Specifically, it establishes a relationship between the rank of a matrix and the fraction of its entries that can be covered by a monochromatic rectangle.

The result states that if the size of the largest monochromatic rectangle in M_f is large, then the rank of the matrix M_f is similarly large, and vice versa. This implies that for a function f with a sufficiently large $\operatorname{Mono}(f)$, the rank of the matrix M_f must also be large.

Step 2: Consider the implication for $\log\left(\frac{1}{\operatorname{Mono}(f)}\right)$ Let us denote the size of the largest monochromatic rectangle in M_f by R_f. We know that $\operatorname{Mono}(f)$ is the fraction of entries in M_f covered by R_f, so we have the inequality

$$\operatorname{Mono}(f) \geq \frac{|R_f|}{|M_f|},$$

where $|R_f|$ is the number of entries in the rectangle R_f, and $|M_f|$ is the total number of entries in M_f. Therefore, we have the bound

$$\log\left(\frac{1}{\mathrm{Mono}(f)}\right) \leq \log\left(\frac{|M_f|}{|R_f|}\right).$$

Step 3: Relating $\log\left(\frac{1}{\mathrm{Mono}(f)}\right)$ to $\mathrm{rank}(f)$ Next, we use the fact that Theorem 2.11 implies that the rank of the matrix M_f is bounded above by a polynomial function of the size of the largest monochromatic rectangle. Specifically, if the largest monochromatic rectangle has size $|R_f|$, then

$$\mathrm{rank}(f) \leq |R_f|^{O(1)}.$$

Thus, we can express the rank of f in terms of the size of the rectangle R_f as

$$\mathrm{rank}(f) = |R_f|^{O(1)}.$$

Step 4: Conclude the proof Using the above relationship, we can now establish the desired inequality. Since $\log\left(\frac{1}{\mathrm{Mono}(f)}\right) \leq \log\left(\frac{|M_f|}{|R_f|}\right)$, and $|M_f|$ is a constant factor relative to $|R_f|$, it follows that

$$\log\left(\frac{1}{\mathrm{Mono}(f)}\right) = (\log \mathrm{rank}(f))^{O(1)}.$$

Thus, we have shown that if Open Problem 2.20 has a positive answer, then the given inequality holds for all functions f. □

D.9.3 Claude

Proof. We prove that if Open Problem 2.20 has a positive answer, then for every function f, we have $\log(1/\mathrm{Mono}(f)) = (\log \mathrm{rank}(f))^{O(1)}$.

Recall that Open Problem 2.20 asks whether $\log \chi(f) = (\log \mathrm{rank}(f))^{O(1)}$ for all functions f, where $\chi(f)$ denotes the chromatic number of the communication matrix M_f.

Assume Open Problem 2.20 has a positive answer. That is, there exist constants $c, d > 0$ such that for all functions f:

$$\log \chi(f) \leq c \cdot (\log \mathrm{rank}(f))^d.$$

Following the proof structure of Theorem 2.11 (Combinatorial Version), we proceed as follows. The chromatic number $\chi(f)$ represents the minimum number of monochromatic rectangles needed to partition the matrix M_f. By definition, if M_f can be partitioned into $\chi(f)$ monochromatic rectangles, then at least one of these rectangles must cover at least a fraction $1/\chi(f)$ of the entries of M_f.

Therefore, we have:

$$\mathrm{Mono}(f) \geq \frac{1}{\chi(f)}.$$

Taking logarithms of both sides:

$$\log \mathrm{Mono}(f) \geq \log\left(\frac{1}{\chi(f)}\right) = -\log \chi(f).$$

Multiplying by -1 and reversing the inequality:

$$\log\left(\frac{1}{\mathrm{Mono}(f)}\right) \leq \log \chi(f).$$

Now, applying our assumption that Open Problem 2.20 has a positive answer:

$$\log\left(\frac{1}{\mathrm{Mono}(f)}\right) \leq \log \chi(f) \leq c \cdot (\log \mathrm{rank}(f))^d.$$

This establishes that:

$$\log\left(\frac{1}{\mathrm{Mono}(f)}\right) = O\left((\log \mathrm{rank}(f))^d\right) = (\log \mathrm{rank}(f))^{O(1)}.$$

Thus, if Open Problem 2.20 receives a positive answer, then for every function f, we have $\log(1/\mathrm{Mono}(f)) = (\log \mathrm{rank}(f))^{O(1)}$, as required. □

D.9.4 Gemini

Proof. Let $D(f)$ denote the deterministic communication complexity of the function f. Open Problem 2.20 refers to the Log-Rank Conjecture, which asserts that $D(f)$ is bounded polynomially by the logarithm of the rank of the communication matrix M_f. Specifically, a positive answer to Open Problem 2.20 implies that there exists a constant $c > 0$ such that for every function f,

$$D(f) = O((\log \operatorname{rank}(f))^c).$$

We aim to bound $\log(1/\mathrm{Mono}(f))$ using this assumption.

Consider an optimal deterministic communication protocol for f. This protocol can be represented by a binary tree $\mathcal{T}$ of depth $d = D(f)$. The leaves of $\mathcal{T}$ correspond to a partition of the domain $X \times Y$ into disjoint combinatorial rectangles $R_1, R_2, \ldots, R_L$, where L is the number of leaves. Since the protocol computes f, every rectangle R_i associated with a leaf is f-monochromatic (i.e., f is constant on all inputs $(x, y) \in R_i$).

The number of leaves in a binary tree of depth d is at most 2^d. Therefore, $L \leq 2^{D(f)}$. Let $\mu(R)$ denote the fraction of entries of M_f covered by a rectangle R. Since the rectangles $\{R_i\}_{i=1}^{L}$ form a partition of the entire domain, the sum of their measures is 1:

$$\sum_{i=1}^{L} \mu(R_i) = 1.$$

By the averaging principle, there must exist at least one rectangle $R_{max} \in \{R_1, \ldots, R_L\}$ such that

$$\mu(R_{max}) \geq \frac{1}{L}.$$

By definition, $\mathrm{Mono}(f)$ is the fraction of entries covered by the largest monochromatic rectangle of f. Since R_{max} is a monochromatic rectangle, we have

$$\mathrm{Mono}(f) \geq \mu(R_{max}) \geq \frac{1}{L} \geq \frac{1}{2^{D(f)}} = 2^{-D(f)}.$$

Taking the logarithm of the reciprocal of this inequality, we obtain

$$\log\left(\frac{1}{\mathrm{Mono}(f)}\right) \leq \log(2^{D(f)}) = D(f).$$

We now apply the assumption that Open Problem 2.20 holds. Substituting the bound $D(f) = (\log \operatorname{rank}(f))^{O(1)}$ into the inequality above yields

$$\log\left(\frac{1}{\mathrm{Mono}(f)}\right) \leq (\log \operatorname{rank}(f))^{O(1)}.$$

This completes the proof. □

D.9.5 Grok

Proof. Assume that Open Problem 2.20 has a positive answer, that is, there exists a universal constant $c \geq 1$ such that for every Boolean function g,

$$D(g) \leq (\log \operatorname{rank}(g))^c,$$

where $\log$ denotes the base-2 logarithm and $\operatorname{rank}(g)$ is the rank of the communication matrix of g over the reals.

Let f be an arbitrary Boolean function and let $r = \operatorname{rank}(f)$. By the assumption,

$$D(f) \leq (\log r)^c.$$

There thus exists a deterministic protocol computing f with communication complexity at most $(\log r)^c$. Any such protocol induces a partition of the input space $X \times Y$ into at most $2^{(\log r)^c}$ monochromatic rectangles (corresponding to the leaves of the protocol tree).

Let $s = |X \times Y|$ denote the total number of entries in the communication matrix M_f. These monochromatic rectangles are pairwise disjoint and their union is $X \times Y$, so their sizes sum to s. It follows that the largest rectangle has size at least

$$\frac{s}{2^{(\log r)^c}}.$$

Therefore,

$$\mathrm{Mono}(f) \geq 2^{-(\log r)^c},$$

which implies

$$\frac{1}{\mathrm{Mono}(f)} \leq 2^{(\log r)^c}$$

and

$$\log\left(\frac{1}{\mathrm{Mono}(f)}\right) \leq (\log r)^c = (\log \mathrm{rank}(f))^c.$$

Since c is a fixed constant independent of f, we have

$$\log\left(\frac{1}{\mathrm{Mono}(f)}\right) = (\log \mathrm{rank}(f))^{O(1)}.$$

□

D.10 Problem 10

D.10.1 Problem Statement

Theorem D.10 (Exercise 2.23)**.** *Let INTER* : $\{0,1\}^n \times \{0,1\}^n \to \{0,1,\ldots,n\}$ *be the function that counts the number entries in which* $x_i = y_i = 1$ *(think of* x *and* y *as sets, then INTER*(x,y) *is the size of their intersection). Show that the rank of* M_{INTER} *is* n*. Conclude that for nonboolean functions the gap between* $D(f)$ *and* $\log$ *rank*(f) *may be exponential.*

Show that the rank of M_{IP} *over* $GF(2)$ *is* n*. Conclude that if we consider rank over finite fields (instead of the reals) the gap between* $D(f)$ *and* $\log$ *rank*(f) *may be exponential.*

D.10.2 GPT

Proof. Let $x = (x_1, x_2, \ldots, x_n), y = (y_1, y_2, \ldots, y_n) \in \{0,1\}^n$ be two binary vectors. The function INTER : $\{0,1\}^n \times \{0,1\}^n \to \{0,1,\ldots,n\}$ is defined by

$$\mathrm{INTER}(x,y) = |\{i \mid x_i = y_i = 1\}|.$$

We aim to show that the rank of M_{INTER}, the matrix representation of the function INTER, is n.

Step 1: Understanding the function and its matrix representation The function INTER(x,y) counts the number of positions i such that both $x_i = 1$ and $y_i = 1$. To formalize this as a matrix, we need to understand the behavior of INTER(x,y) for all pairs of binary vectors x and y.

The matrix M_{INTER} is a matrix whose rows are indexed by the binary vectors in $\{0,1\}^n$ and whose columns correspond to the values of INTER(x,y) for each pair (x,y). Since INTER(x,y) is a sum over the positions where both x_i and y_i are 1, the function is linear in both x and y.

Step 2: Linear dependence in the rows of M_{INTER} Consider the set of all binary vectors of length n. Denote by e_i the standard basis vector in $\{0,1\}^n$ with 1 in the i-th position and 0 elsewhere. We claim that the rows of M_{INTER} corresponding to the vectors x and y are linearly independent.

Each row of M_{INTER} is a vector representing the outcome of the function INTER(x,y) for a fixed x. The vector INTER$(x,\cdot)$ is linear in the components of x, meaning the set of rows is spanned by the vectors corresponding to $x = e_1, e_2, \ldots, e_n$. Since the vectors $e_1, e_2, \ldots, e_n$ are linearly independent, we conclude that the rank of M_{INTER} is at most n.

Step 3: Showing that the rank is exactly n To show that the rank is exactly n, we observe that for any binary vector x, the function INTER$(x, \cdot)$ is a sum of the indicator functions of the positions where $x_i = 1$. These indicator functions form a linearly independent set of functions, and thus the rank of M_{INTER} is exactly n.

Step 4: Conclusion for nonboolean functions Since the rank of M_{INTER} is n, we conclude that the gap between the deterministic query complexity $D(f)$ of a function f and the logarithm of its rank, $\log \text{rank}(f)$, may be exponential for nonboolean functions.

Next, consider the case where we look at the rank of the function over a finite field $GF(2)$, which is the same field over which the function INTER is defined. Over $GF(2)$, the function INTER(x, y) is also linear, and its matrix representation M_{IP} has the same structure as M_{INTER}, since the binary operations in $GF(2)$ correspond to the same logical operations as in $\{0, 1\}$. Thus, the rank of M_{IP} over $GF(2)$ is also n.

Step 5: Conclusion for rank over finite fields Since the rank of M_{IP} over $GF(2)$ is n, we conclude that if we consider the rank over finite fields, the gap between the deterministic query complexity $D(f)$ and $\log \text{rank}(f)$ may be exponential as well.

This completes the proof. □

D.10.3 Claude

Proof. We prove both parts of the theorem.

Part 1: Rank of M_{INTER} over the reals.

The communication matrix M_{INTER} is a $2^n \times 2^n$ matrix where the entry in row x and column y (for $x, y \in \{0, 1\}^n$) is INTER$(x, y) = \sum_{i=1}^n x_i y_i$.

We can write INTER$(x, y) = \sum_{i=1}^n x_i y_i = \langle x, y \rangle$, where $\langle \cdot, \cdot \rangle$ denotes the standard inner product. This gives us

$$M_{\text{INTER}} = \sum_{i=1}^{n} M_i$$

where M_i is the matrix with $(M_i)_{x,y} = x_i y_i$. Note that M_i is the rank-1 matrix that can be written as $M_i = u_i v_i^T$, where u_i and v_i are the vectors indexed by $x, y \in \{0, 1\}^n$ with $(u_i)_x = x_i$ and $(v_i)_y = y_i$.

Since M_{INTER} is a sum of n rank-1 matrices, we have rank$(M_{\text{INTER}}) \leq n$.

To show that rank$(M_{\text{INTER}}) = n$, we exhibit n linearly independent rows. Consider the rows corresponding to the standard basis vectors $e_1, e_2, \ldots, e_n \in \{0, 1\}^n$, where e_i has a 1 in position i and 0s elsewhere. For row e_i, the entry in column y is INTER$(e_i, y) = y_i$.

The $n \times 2^n$ submatrix formed by these rows has row i equal to the vector $(y_i)_{y \in \{0,1\}^n}$. These n rows are linearly independent: if $\sum_{i=1}^n c_i (y_i)_{y \in \{0,1\}^n} = 0$ for all y, then evaluating at $y = e_j$ gives $c_j = 0$ for all j.

Therefore, rank$(M_{\text{INTER}}) = n$.

Since $D(\text{INTER}) = n + 1$ (Alice and Bob can each send their n-bit strings, or use a more efficient protocol requiring $O(n)$ bits), while $\log \text{rank}(M_{\text{INTER}}) = \log n$, we have an exponential gap of $\Omega(n/\log n)$ between $D(f)$ and $\log \text{rank}(f)$ for nonboolean functions.

Part 2: Rank of M_{IP} over $GF(2)$.

Let IP : $\{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}$ be the inner product function modulo 2, defined by IP$(x, y) = \sum_{i=1}^n x_i y_i \pmod 2$.

Over $GF(2)$, we have

$$M_{\text{IP}} = \sum_{i=1}^{n} M_i \pmod 2$$

where M_i is the rank-1 matrix over $GF(2)$ with $(M_i)_{x,y} = x_i y_i$.

As before, rank$_{GF(2)}(M_{\text{IP}}) \leq n$.

To show equality, consider the rows corresponding to $e_1, \ldots, e_n$. For row e_i, the entry in column y is IP$(e_i, y) = y_i \pmod 2$. The $n \times 2^n$ submatrix formed by these rows has row i equal to $(y_i)_{y \in \{0,1\}^n}$ over $GF(2)$.

These rows are linearly independent over $GF(2)$: if $\sum_{i=1}^{n} c_i(y_i)_y = 0$ over $GF(2)$ for all y, then evaluating at $y = e_j$ gives $c_j = 0$ for all $j \in \{1, \ldots, n\}$.

Therefore, $\text{rank}_{GF(2)}(M_{\text{IP}}) = n$.

Since $D(\text{IP}) = n + 1$ (in the worst case, Alice and Bob must exchange $\Omega(n)$ bits), while $\log \text{rank}_{GF(2)}(M_{\text{IP}}) = \log n$, we have an exponential gap between $D(f)$ and $\log \text{rank}_{GF(2)}(f)$. □

D.10.4 Gemini

Proof. We first analyze the rank of the matrix M_{INTER} associated with the function $\text{INTER} : \{0,1\}^n \times \{0,1\}^n \to \{0, 1, \ldots, n\}$, defined by $\text{INTER}(x, y) = \sum_{i=1}^{n} x_i y_i$, where the sum is over the real numbers. The matrix M_{INTER} is a $2^n \times 2^n$ matrix with entries indexed by $x, y \in \{0,1\}^n$.

Let A be a $2^n \times n$ matrix where the row indexed by $x \in \{0,1\}^n$ is the vector x^T. Similarly, let B be a $2^n \times n$ matrix where the row indexed by $y \in \{0,1\}^n$ is the vector y^T. The entry (x, y) of the matrix product AB^T is given by the standard inner product of the vectors x and y:

$$(AB^T)_{x,y} = \sum_{k=1}^{n} A_{x,k}(B^T)_{k,y} = \sum_{k=1}^{n} x_k y_k = \text{INTER}(x, y).$$

Thus, $M_{\text{INTER}} = AB^T$. Since M_{INTER} is the product of a $2^n \times n$ matrix and an $n \times 2^n$ matrix, its rank is bounded by the inner dimension. Therefore, $\text{rank}(M_{\text{INTER}}) \leq n$.

To establish a lower bound, consider the submatrix of M_{INTER} defined by the rows and columns indexed by the standard basis vectors $\{e_1, e_2, \ldots, e_n\}$, where e_i has a 1 in the i-th position and 0 elsewhere. The entry corresponding to row e_i and column e_j is

$$\text{INTER}(e_i, e_j) = e_i^T e_j = \delta_{ij},$$

where δ_{ij} is the Kronecker delta. This submatrix is the $n \times n$ identity matrix I_n. Since the rank of a matrix is at least the rank of any of its submatrices, and $\text{rank}(I_n) = n$, we have $\text{rank}(M_{\text{INTER}}) \geq n$. Combining the upper and lower bounds, we conclude that $\text{rank}(M_{\text{INTER}}) = n$.

We now consider the relationship between $D(\text{INTER})$ and $\log \text{rank}(M_{\text{INTER}})$. The deterministic communication complexity of the set intersection problem is lower-bounded by the complexity of the Disjointness function, $\text{DISJ}(x, y)$, which is 1 if $\text{INTER}(x, y) = 0$ and 0 otherwise. It is a well-known result that $D(\text{DISJ}) \geq n$. Since any protocol computing $\text{INTER}(x, y)$ can determine $\text{DISJ}(x, y)$ with no additional communication, we have $D(\text{INTER}) \geq n$. However, $\log \text{rank}(M_{\text{INTER}}) = \log n$. The gap between $D(\text{INTER}) \approx n$ and $\log \text{rank}(M_{\text{INTER}}) \approx \log n$ is exponential. This shows that for non-boolean functions (specifically those with range size polynomial in n), the logarithm of the rank is not necessarily polynomially related to the deterministic communication complexity.

Next, we consider the Inner Product function $\text{IP}(x, y) = \sum_{i=1}^{n} x_i y_i \pmod 2$. The communication matrix M_{IP} has entries in the field $GF(2)$. We analyze the rank of this matrix over $GF(2)$. Similar to the previous case, let A and B be $2^n \times n$ matrices containing the input vectors x and y as rows, respectively. Over the field $GF(2)$, the matrix product AB^T yields entries

$$(AB^T)_{x,y} = \sum_{k=1}^{n} x_k y_k \pmod 2 = \text{IP}(x, y).$$

Thus, $M_{\text{IP}} = AB^T$ over $GF(2)$. The rank of M_{IP} over $GF(2)$ is bounded by the inner dimension n. Hence, $\text{rank}_{GF(2)}(M_{\text{IP}}) \leq n$.

Consider the submatrix indexed by the standard basis vectors $\{e_1, \ldots, e_n\}$. The entry (e_i, e_j) is $e_i^T e_j \pmod 2$, which is equal to δ_{ij}. This submatrix is the identity matrix I_n, which has full rank n over $GF(2)$. Therefore, $\text{rank}_{GF(2)}(M_{\text{IP}}) \geq n$. We conclude that $\text{rank}_{GF(2)}(M_{\text{IP}}) = n$.

Regarding the communication complexity, it is a standard result that the deterministic communication complexity of the Inner Product function is $D(\text{IP}) = n$ (Alice must send her entire input to Bob, or vice versa, to compute the parity of the intersection exactly). Comparing the complexity to the log-rank quantity, we have $D(\text{IP}) = n$ while $\log \text{rank}_{GF(2)}(M_{\text{IP}}) = \log n$. This demonstrates an exponential gap

between the deterministic communication complexity and the logarithm of the rank when the rank is computed over a finite field. This implies that the Log-Rank Conjecture, which applies to boolean matrices over the reals, does not generalize to rank over finite fields. □

D.10.5 Grok

Proof. We first prove that the rank of M_{INTER} (over $\mathbb{R}$) is n.

The matrix M_{INTER} has rows and columns indexed by elements of $\{0,1\}^n$, with $(M_{\text{INTER}})_{x,y} = \sum_{i=1}^n x_i y_i$.

Let $e^1, \ldots, e^n \in \{0,1\}^n$ denote the standard basis vectors (with 1 in the j-th position for e^j). For each $j = 1, \ldots, n$, let r^j denote the row of M_{INTER} corresponding to input e^j. Then $(r^j)_y = y_j$ for all $y \in \{0,1\}^n$.

For arbitrary $x \in \{0,1\}^n$, we have $x = \sum_{j:x_j=1} e^j$ (as vectors in $\mathbb{R}^n$), and linearity of the dot product yields that the row corresponding to x is $\sum_{j:x_j=1} r^j$. Thus, the row space of M_{INTER} is contained in the span of $\{r^1, \ldots, r^n\}$.

To show linear independence of $\{r^1, \ldots, r^n\}$, suppose $\sum_{j=1}^n c_j r^j = 0$ with $c_j \in \mathbb{R}$. Then, for all $y \in \{0,1\}^n$, $\sum_{j=1}^n c_j y_j = 0$. Setting $y = e^k$ gives $c_k = 0$ for each k. Hence $\{r^1, \ldots, r^n\}$ are linearly independent, the row space has dimension n, and $\operatorname{rank}(M_{\text{INTER}}) = n$.

Alice can send her input x (using n bits), allowing Bob to compute $\text{INTER}(x, y)$; thus $D(\text{INTER}) \leq n$. Since $\log \operatorname{rank}(M_{\text{INTER}}) = O(\log n)$ and $n = 2^{O(\log n)} = 2^{O(\log \operatorname{rank}(M_{\text{INTER}}))}$, the gap between $D(\text{INTER})$ and $\log \operatorname{rank}(M_{\text{INTER}})$ is exponential for this non-Boolean function.

Next, we prove that the rank of M_{IP} over $\text{GF}(2)$ is n.

The matrix M_{IP} has $(M_{\text{IP}})_{x,y} = \sum_{i=1}^n x_i y_i \pmod 2$. The preceding notation for e^j and r^j applies identically, with $(r^j)_y = y_j$.

For arbitrary $x \in \{0,1\}^n$, we have $x = \sum_{j:x_j=1} e^j$ in $(\text{GF}(2))^n$, and linearity over $\text{GF}(2)$ yields that the row corresponding to x is $\sum_{j:x_j=1} r^j$ (vector addition in $(\text{GF}(2))^{\{0,1\}^n}$). Thus, the row space over $\text{GF}(2)$ is contained in the span of $\{r^1, \ldots, r^n\}$.

To show linear independence over $\text{GF}(2)$, suppose $\sum_{j=1}^n c_j r^j = 0$ with $c_j \in \text{GF}(2)$. Then, for all $y \in \{0,1\}^n$, $\sum_{j=1}^n c_j y_j = 0$ in $\text{GF}(2)$. Setting $y = e^k$ gives $c_k = 0$ for each k. Hence $\{r^1, \ldots, r^n\}$ are linearly independent over $\text{GF}(2)$, the row space has dimension n, and the rank of M_{IP} over $\text{GF}(2)$ is n.

The (Boolean) inner product function has deterministic communication complexity $\Omega(n)$, while $\log \operatorname{rank}_{\text{GF}(2)}(M_{\text{IP}}) = O(\log n)$. Thus $D(\text{IP}) = 2^{O(\log \operatorname{rank}_{\text{GF}(2)}(M_{\text{IP}}))}$, so the gap between $D(f)$ and $\log \operatorname{rank}_{\text{GF}(2)}(f)$ is exponential when rank is taken over finite fields.

□

E Problems in Chapter 3

E.1 Problem 1

E.1.1 Problem Statement

Theorem E.1. *The following are basic properties of the definitions given above:*

For $0 < \varepsilon < \varepsilon' < 1/2$, $R_\varepsilon(f) \leq O(\log_{\varepsilon'} \varepsilon \cdot R_{\varepsilon'}(f))$. Conclude that the error probability can be reduced with a small penalty in the communication complexity. Hint: Start by proving the same relation with respect to $R^1_\varepsilon(f)$. Then use Chernoff inequality to generalize your proof for $R(f)$.

$R_\varepsilon(f) \leq R^1_\varepsilon(f) \leq O(\log \varepsilon^{-1}) R_0(f)$.

$R_0(f) = O(\max[R^1(f), R^1(not(f))])$.

E.1.2 GPT

Proof. We begin by proving the relation for $R^1_\varepsilon(f)$. By definition, for a given ε, the quantity $R^1_\varepsilon(f)$ represents the communication complexity associated with the function f with an error probability of at most ε.

First, we observe that the function $R_\varepsilon(f)$ is defined as the minimum communication complexity of f with an error probability less than or equal to ε. Thus, we have the inequality

$$R_\varepsilon(f) \leq R^1_\varepsilon(f).$$

Now, we proceed to establish the upper bound on $R^1_\varepsilon(f)$ in terms of $R_{\varepsilon'}(f)$. By the given conditions, for $0 < \varepsilon < \varepsilon' < 1/2$, we know that

$$R_\varepsilon(f) \leq O(\log_{\varepsilon'} \varepsilon \cdot R_{\varepsilon'}(f)).$$

We can rewrite the logarithmic term using the change of base formula for logarithms:

$$\log_{\varepsilon'} \varepsilon = \frac{\log \varepsilon}{\log \varepsilon'}.$$

Thus, the inequality becomes

$$R_\varepsilon(f) \leq O\left(\frac{\log \varepsilon}{\log \varepsilon'} \cdot R_{\varepsilon'}(f)\right).$$

Since $R^1_\varepsilon(f)$ represents a more refined quantity than $R_\varepsilon(f)$, we conclude that

$$R^1_\varepsilon(f) \leq O\left(\frac{\log \varepsilon}{\log \varepsilon'} \cdot R_{\varepsilon'}(f)\right).$$

Next, we use the Chernoff inequality to generalize this result for the quantity $R(f)$. The Chernoff inequality provides an exponential bound on the probability of error, and since the error probability is reduced with a small penalty in communication complexity, we apply the inequality to extend the result to $R(f)$. In particular, for any $\delta > 0$, the Chernoff inequality guarantees that

$$P(\text{error}) \leq e^{-\delta^2 n/2}.$$

Thus, using Chernoff bounds, we conclude that

$$R(f) \leq O(\log \varepsilon^{-1}) R_0(f).$$

Finally, we observe the relationship between $R_0(f)$ and the other quantities. By definition,

$$R_0(f) = O(\max[R^1(f), R^1(\neg f)]).$$

This completes the proof of the desired result. □

E.1.3 Claude

Proof. We prove each of the three properties in turn.

Property 1: For $0 < \varepsilon < \varepsilon' < 1/2$, we have $R_\varepsilon(f) \leq O(\log_{\varepsilon'} \varepsilon \cdot R_{\varepsilon'}(f))$.

We first establish the relation for one-way communication complexity. Let Π' be a protocol computing f with error at most ε' and communication cost $R^1_{\varepsilon'}(f)$. We run Π' independently k times and take the majority vote of the outputs. Let X_i be the indicator random variable that equals 1 if the i-th execution gives an incorrect answer. Then $\mathbb{E}[X_i] \leq \varepsilon'$ and the X_i are independent. Let $X = \sum_{i=1}^k X_i$. The majority vote is incorrect when $X > k/2$.

By the Chernoff bound, for $\delta > 0$ such that $(1+\delta)\varepsilon' = 1/2$, we have

$$\Pr[X > k/2] = \Pr[X > (1+\delta)k\varepsilon'] \leq \exp\left(-\frac{\delta^2 k \varepsilon'}{3}\right).$$

Since $\varepsilon' < 1/2$, we have $\delta = \frac{1-2\varepsilon'}{2\varepsilon'} > 0$. To achieve error probability at most ε, we require

$$\exp\left(-\frac{\delta^2 k \varepsilon'}{3}\right) \leq \varepsilon.$$

Taking logarithms, we need $k \geq \frac{3\ln(\varepsilon^{-1})}{\delta^2\varepsilon'}$. Since $\delta^2 = \Theta(1)$ for fixed ε', we have $k = O(\log(\varepsilon^{-1}))$. More precisely, $k = O\left(\frac{\ln(\varepsilon^{-1})}{\varepsilon'}\right)$.

For $\varepsilon < \varepsilon'$, we can write $\log_{\varepsilon'} \varepsilon = \frac{\ln \varepsilon}{\ln \varepsilon'} = \frac{\ln(\varepsilon^{-1})}{\ln(\varepsilon'^{-1})}$, so $k = O(\log_{\varepsilon'} \varepsilon)$.

The total communication is $k \cdot R^1_{\varepsilon'}(f) = O(\log_{\varepsilon'} \varepsilon \cdot R^1_{\varepsilon'}(f))$, which gives $R^1_\varepsilon(f) \leq O(\log_{\varepsilon'} \varepsilon \cdot R^1_{\varepsilon'}(f))$.

For general (two-way) protocols, the same argument applies since we can repeat any protocol Π' with error ε' independently k times and take majority vote. Thus $R_\varepsilon(f) \leq O(\log_{\varepsilon'} \varepsilon \cdot R_{\varepsilon'}(f))$.

Property 2: $R_\varepsilon(f) \leq R^1_\varepsilon(f) \leq O(\log \varepsilon^{-1}) R_0(f)$.

The first inequality $R_\varepsilon(f) \leq R^1_\varepsilon(f)$ is immediate since one-way protocols are a special case of general protocols.

For the second inequality, let Π be a zero-error protocol for f with communication cost $R_0(f)$. By Property 1 applied with $\varepsilon' = 1/3$ (a constant), we have $R^1_\varepsilon(f) \leq O(\log_{1/3} \varepsilon) \cdot R^1_{1/3}(f)$. Since $\log_{1/3} \varepsilon = \frac{\ln \varepsilon}{\ln(1/3)} = O(\ln \varepsilon^{-1})$, we get $R^1_\varepsilon(f) = O(\log \varepsilon^{-1}) \cdot R^1_{1/3}(f)$.

Now, any zero-error protocol can be converted to a one-way protocol with constant error by having Alice send her entire view (which determines the protocol's output). This requires at most $R_0(f)$ bits of communication. Therefore $R^1_{1/3}(f) \leq R_0(f)$, giving us $R^1_\varepsilon(f) \leq O(\log \varepsilon^{-1}) R_0(f)$.

Property 3: $R_0(f) = O(\max[R^1(f), R^1(\neg f)])$.

Let $c = \max[R^1(f), R^1(\neg f)]$. There exist one-way zero-error protocols Π_f and $\Pi_{\neg f}$ for f and $\neg f$ respectively, each using at most c bits.

We construct a zero-error protocol for f as follows: Alice sends both messages from Π_f and $\Pi_{\neg f}$, using at most $2c$ bits. Bob computes both outputs. Since Π_f is a zero-error protocol for f and $\Pi_{\neg f}$ is a zero-error protocol for $\neg f$, exactly one of them outputs 1 on any input (x, y). Bob outputs 1 if Π_f outputs 1, and 0 otherwise. This protocol computes f with zero error using $O(c)$ communication, so $R_0(f) = O(\max[R^1(f), R^1(\neg f)])$. □

E.1.4 Gemini

Proof. We first address the error reduction property for one-sided error. Let $\mathcal{P}'$ be a randomized protocol for f with one-sided error $\varepsilon' < 1/2$ and communication complexity $R^1_{\varepsilon'}(f)$. This implies that for any input (x, y), if $f(x, y) = 0$, $\mathcal{P}'$ outputs 0 with probability 1; if $f(x, y) = 1$, $\mathcal{P}'$ outputs 1 with probability at least $1 - \varepsilon'$.

Construct a new protocol $\mathcal{P}$ that runs $\mathcal{P}'$ independently k times. $\mathcal{P}$ outputs 1 if and only if at least one of the k runs outputs 1; otherwise, it outputs 0. The cost of $\mathcal{P}$ is $k \cdot R^1_{\varepsilon'}(f)$. If $f(x, y) = 0$, all k runs output 0 with probability 1, so $\mathcal{P}$ outputs 0 with probability 1 (error is 0). If $f(x, y) = 1$, the probability that a single run outputs 0 is at most ε'. Since the runs are independent, the probability that all k runs output 0 is at most $(\varepsilon')^k$. We require this error to be at most ε:

$$(\varepsilon')^k \leq \varepsilon \iff k \ln \varepsilon' \leq \ln \varepsilon \iff k \geq \frac{\ln \varepsilon}{\ln \varepsilon'} = \log_{\varepsilon'} \varepsilon.$$

Setting $k = \lceil \log_{\varepsilon'} \varepsilon \rceil$, we obtain $R^1_\varepsilon(f) \leq \lceil \log_{\varepsilon'} \varepsilon \rceil R^1_{\varepsilon'}(f)$.

For the two-sided error case $R_\varepsilon(f)$, let $\mathcal{Q}'$ be a protocol with error $\varepsilon' < 1/2$. Run $\mathcal{Q}'$ independently k times and output the majority result. Let X_i be an indicator variable that is 1 if the i-th run is correct and 0 otherwise. Then $\Pr[X_i = 1] \geq 1 - \varepsilon' > 1/2$. By the Chernoff bound, the probability that the majority is incorrect (i.e., $\sum X_i \leq k/2$) decreases exponentially with k. Specifically, for the error to be bounded by ε, k must be proportional to $\log(1/\varepsilon)$. Since $\log_{\varepsilon'} \varepsilon = \frac{\ln(1/\varepsilon)}{\ln(1/\varepsilon')}$, the scaling factor is of the same logarithmic order. Thus, $R_\varepsilon(f) \leq O(\log_{\varepsilon'} \varepsilon \cdot R_{\varepsilon'}(f))$. This demonstrates that error probability can be reduced to any ε with a multiplicative penalty logarithmic in $1/\varepsilon$.

Next, we establish the inequality chain $R_\varepsilon(f) \leq R^1_\varepsilon(f) \leq O(\log \varepsilon^{-1}) R_0(f)$. The first inequality holds by definition: any protocol with one-sided error ε on 1-inputs and zero error on 0-inputs is a valid two-sided error protocol with error at most ε. For the second inequality, observe that a zero-error protocol (Las Vegas) never errs ($R_0(f)$). This trivially satisfies the conditions of a one-sided error protocol for any $\varepsilon \geq 0$. The factor $O(\log \varepsilon^{-1})$ serves as a loose upper bound derived from the standard relation where a

Las Vegas protocol can be converted to a Monte Carlo protocol (though the reverse direction typically incurs the logarithmic cost). Formally, since $R_0(f) \geq R^1_\varepsilon(f)$ holds for any ε, the bound holds.

Finally, we prove $R_0(f) = O(\max[R^1(f), R^1(\neg f)])$, where $R^1(f)$ denotes one-sided complexity with a constant error (e.g., $1/3$). Let Π_1 be a one-sided error protocol for f with error $\leq 1/2$ (errs only on 1-inputs). Let Π_0 be a one-sided error protocol for $\neg f$ with error $\leq 1/2$ (errs only on 0-inputs of f). Construct a zero-error protocol Π as follows: In each round, Alice and Bob run both Π_1 and Π_0.

- If Π_1 outputs 1, then $f(x, y)$ must be 1 (since Π_1 has 0-sided error on 0-inputs). Terminate and output 1.
- If Π_0 outputs 1 (indicating $\neg f$ is true), then $f(x, y)$ must be 0. Terminate and output 0.
- If both output 0, the result is inconclusive. Repeat the round.

In any round, the correct protocol will output 1 with probability at least $1/2$. Thus, the probability of termination in a given round is at least $1/2$. The expected number of rounds is at most 2. The communication cost per round is $R^1(f) + R^1(\neg f)$. Therefore, the expected complexity is $R_0(f) \leq 2(R^1(f) + R^1(\neg f))$, which implies $R_0(f) = O(\max[R^1(f), R^1(\neg f)])$. □

E.1.5 Grok

Proof. We prove the three properties in order.

First, consider $0 < \varepsilon < \varepsilon' < 1/2$. We show $R_\varepsilon(f) = O(\log_{\varepsilon'} \varepsilon \cdot R_{\varepsilon'}(f))$.

Following the hint, we begin with the one-sided complexity $R^1_\varepsilon(f)$, defined as the minimal communication complexity of a public-coin randomized protocol that, on inputs with $f(x, y) = 0$, always outputs 0, and on inputs with $f(x, y) = 1$, outputs 1 with probability at least $1 - \varepsilon$.

Let Π' be such a protocol with error at most ε' and complexity $R^1_{\varepsilon'}(f)$. Set $k = \lceil \log_{\varepsilon'} \varepsilon \rceil + 1$, so $\varepsilon'^k < \varepsilon$. Using public randomness, generate k independent random strings and run Π' sequentially k times with these strings. Output 1 if at least one run outputs 1, and 0 otherwise.

The communication complexity is at most $k \cdot R^1_{\varepsilon'}(f) = O(\log_{\varepsilon'} \varepsilon \cdot R^1_{\varepsilon'}(f))$. On inputs with $f(x, y) = 0$, every run outputs 0, so the protocol outputs 0. On inputs with $f(x, y) = 1$, the probability that all runs output 0 is at most $\varepsilon'^k < \varepsilon$, so the protocol outputs 1 with probability greater than $1 - \varepsilon$. Thus $R^1_\varepsilon(f) = O(\log_{\varepsilon'} \varepsilon \cdot R^1_{\varepsilon'}(f))$.

To generalize to the two-sided complexity $R_\varepsilon(f)$, let Π' be a protocol with two-sided error at most ε' and complexity $R_{\varepsilon'}(f)$. Choose k odd and sufficiently large (as bounded below). Generate k independent random strings publicly and run Π' sequentially k times, obtaining outputs $b_1, \ldots, b_k$. Output the majority value.

The communication is at most $k \cdot R_{\varepsilon'}(f)$. For any fixed input with correct value b^*, the probability a single $b_i \neq b^*$ is at most ε'. Let Z be the number of incorrect b_i; then $\mathbb{E}[Z] \leq \varepsilon' k$. The output is incorrect if $Z > k/2$.

By the Chernoff bound,

$$\Pr[Z > k/2] \leq \exp\left(-2k\left(1/2 - \varepsilon'\right)^2\right),$$

since $k/2 = \varepsilon' k + k(1/2 - \varepsilon')$ and the deviation is $t = k(1/2 - \varepsilon')$. Setting

$$k \geq \frac{\ln(1/\varepsilon)}{2(1/2 - \varepsilon')^2}$$

ensures the error probability is at most ε. Thus $k = O(\ln(1/\varepsilon)/(1/2 - \varepsilon')^2)$. Since $\ln(1/\varepsilon) = \log_{\varepsilon'} \varepsilon \cdot \ln(1/\varepsilon')$ and both $\ln(1/\varepsilon')$ and $(1/2 - \varepsilon')^2$ depend only on ε', we have $k = O(\log_{\varepsilon'} \varepsilon)$ with the implied constant depending on ε'. Therefore $R_\varepsilon(f) = O(\log_{\varepsilon'} \varepsilon \cdot R_{\varepsilon'}(f))$.

This shows that the error probability can be reduced from ε' to an arbitrary smaller ε at the cost of an $O(\log_{\varepsilon'} \varepsilon)$ factor in communication complexity, which is a small penalty when the original complexity is large.

Next, we prove $R_\varepsilon(f) \le R^1_\varepsilon(f) \le O(\log \varepsilon^{-1}) R_0(f)$, where $R_0(f)$ denotes the Las Vegas (zero-error randomized) communication complexity, defined as the minimal expected communication over always-correct public-coin protocols.

The inequality $R_\varepsilon(f) \le R^1_\varepsilon(f)$ holds because any one-sided protocol with error at most ε (error 0 on 0-inputs, at most ε on 1-inputs) is a valid two-sided protocol with error at most ε.

For $R^1_\varepsilon(f) \le O(\log \varepsilon^{-1}) R_0(f)$, let Π be a Las Vegas protocol computing f with expected communication $R_0(f)$. From the third property (proved below), such a Π can be constructed with a constant success probability per round $p \ge 1/6$ (say). The number of rounds until halting is geometric with parameter p, so $\Pr[\text{rounds} > m] \le (1-p)^m \le e^{-pm}$.

Construct a protocol running Π for at most $m = \lceil (1/p) \ln(1/\varepsilon) \rceil + 1$ rounds (so $\Pr[\text{not halt}] < \varepsilon$). If it halts, output the value; otherwise, output 0.

The communication is at most $O(m \cdot R_0(f)) = O(\log \varepsilon^{-1} \cdot R_0(f))$. On 0-inputs, if it halts the output is correct (0), and if not the default is 0, so always correct. On 1-inputs, if it halts the output is correct (1), and if not the default 0 is incorrect with probability less than ε. Thus the protocol has one-sided error at most ε.

Finally, we prove $R_0(f) = O(\max[R^1(f), R^1(\neg f)])$, where here $R^1(f) = R^1_{1/3}(f)$ (constant error $1/3$).

Let $c = \max(R^1(f), R^1(\neg f))$. Let Π be a one-sided protocol for f with error at most $1/3$ (so $\Pr[\text{output } 1 \mid f = 1] \ge 2/3$, always output 0 if $f = 0$), complexity at most c. Let Σ be the analogous protocol for $\neg f$.

Construct the following Las Vegas protocol: In each round, use public randomness to choose Π or Σ with equal probability $1/2$ and run it. If the chosen protocol outputs 1, halt and output 1 if it was Π, or 0 if it was Σ. Otherwise, continue.

When the protocol halts, the output is correct (no false positives in either verifier). On inputs with $f = 1$, Π accepts with probability $\ge 2/3$ and Σ always rejects, so the probability of halting in a round is at least $(1/2)(2/3) = 1/3$. Similarly on $f = 0$. Thus the expected number of rounds is at most 3, and the expected communication is $O(c) = O(\max[R^1(f), R^1(\neg f)])$.

□

E.2 Problem 2

E.2.1 Problem Statement

Theorem E.2 (Exercise 3.6)**.** *The following protocol for equality testing achieves similar performance to the standard equality protocol. Alice and Bob view their inputs a and b as n-bit integers (between 1 and 2^n). Alice chooses a prime number p at random among the first n^2 primes. She sends both p and $a \bmod p$ to Bob. Bob checks whether $a \bmod p = b \bmod p$, and if so he outputs 1, otherwise he outputs 0.*

E.2.2 GPT

Proof. Let Alice and Bob view their inputs a and b as n-bit integers, where $a, b \in \{1, 2, \ldots, 2^n\}$. Alice chooses a prime number p at random among the first n^2 primes and sends both p and $a \bmod p$ to Bob. Bob checks whether $a \bmod p = b \bmod p$ and outputs 1 if they are equal, otherwise outputs 0.

We aim to show that this protocol achieves similar performance to the standard equality testing protocol, which simply compares a and b directly. The standard equality testing protocol outputs 1 if $a = b$, and outputs 0 otherwise.

Correctness of the Protocol

First, we consider the case where $a = b$. In this case, for any prime p, we have

$$a \bmod p = b \bmod p.$$

Thus, Bob will receive the same value for $a \bmod p$ and $b \bmod p$, and consequently, Bob will output 1.

Next, we consider the case where $a \neq b$. In this case, we wish to show that with high probability, Bob will output 0. Since a and b are distinct, there exists at least one prime p such that

$$a \bmod p \neq b \bmod p.$$

To analyze the probability that $a \bmod p = b \bmod p$, observe that for any fixed prime p, the values of $a \bmod p$ and $b \bmod p$ are independent and uniformly distributed in the set $\{0, 1, \ldots, p-1\}$. The probability that $a \bmod p = b \bmod p$ is therefore

$$\frac{1}{p}.$$

Thus, for a randomly chosen prime p, the probability that $a \bmod p = b \bmod p$ is at most $1/p$. Since p is chosen from the first n^2 primes, the largest prime in this set is approximately $n^2 \log n$. Therefore, the probability that $a \bmod p = b \bmod p$ for the largest prime is at most $1/(n^2 \log n)$.

Probability of a False Positive

The protocol succeeds in outputting 1 when $a = b$, and fails when $a \neq b$. To bound the probability of a false positive (i.e., the case where $a \neq b$ but Bob outputs 1), we use the union bound. The false positive occurs if there exists a prime p such that $a \bmod p = b \bmod p$. The probability of this event for a fixed prime p is at most $1/p$. Summing over all primes p in the first n^2 primes, we obtain the following upper bound on the probability of a false positive:

$$\sum_{i=1}^{n^2} \frac{1}{p_i},$$

where p_i denotes the i-th prime. Using the approximation for the sum of reciprocals of the first n^2 primes, we have

$$\sum_{i=1}^{n^2} \frac{1}{p_i} \sim \log n^2 = 2 \log n.$$

Thus, the probability of a false positive is at most $O(\log n / n)$.

Conclusion

We have shown that the protocol is correct in the sense that it outputs 1 when $a = b$, and outputs 0 with high probability when $a \neq b$. The probability of a false positive is $O(\log n / n)$, which is small for large n. Therefore, the protocol achieves similar performance to the standard equality protocol. □

E.2.3 Claude

Proof. We analyze the correctness and communication complexity of the proposed protocol.

Completeness: If $a = b$, then for any prime p, we have $a \bmod p = b \bmod p$. Therefore, Bob always outputs 1 when $a = b$, giving perfect completeness.

Soundness: Suppose $a \neq b$. We must bound the probability that Bob incorrectly outputs 1. Bob outputs 1 if and only if $a \equiv b \pmod{p}$, which occurs precisely when p divides $|a - b|$.

Since $a, b \in \{1, 2, \ldots, 2^n\}$ and $a \neq b$, we have $1 \leq |a - b| < 2^n$. By the fundamental theorem of arithmetic, $|a - b|$ has at most n distinct prime divisors (since if $|a - b|$ had more than n distinct prime divisors, each at least 2, then $|a - b| \geq 2^{n+1} > 2^n$, a contradiction).

Let p_k denote the k-th prime number. By the prime number theorem, the n^2-th prime satisfies $p_{n^2} \sim n^2 \ln(n^2) = 2n^2 \ln n$. More precisely, for sufficiently large n, we have $p_{n^2} \geq n^2 \ln n$.

The probability that a randomly chosen prime p from the first n^2 primes divides $|a - b|$ is at most $\frac{n}{n^2} = \frac{1}{n}$, since $|a - b|$ has at most n distinct prime divisors and there are n^2 primes to choose from.

Therefore, the error probability is at most $\frac{1}{n}$, which can be made arbitrarily small by choosing n large enough or by repeating the protocol with independent random primes.

Communication Complexity: Alice sends the prime p and the value $a \bmod p$ to Bob. Since $p \leq p_{n^2} = O(n^2 \ln n)$, we can represent p using $O(\log(n^2 \ln n)) = O(\log n)$ bits. The value $a \bmod p$ requires at most $O(\log p) = O(\log n)$ bits. Thus, the total communication is $O(\log n)$ bits.

This matches the performance of the standard equality protocol, which uses $O(\log n)$ bits of communication with error probability $O(1/n)$. □

E.2.4 Gemini

Proof. We analyze the performance of the described protocol by establishing its communication complexity and its error probability. Let the inputs a and b be distinct integers in the range $\{1, \ldots, 2^n\}$.

First, we determine the communication complexity. The protocol requires Alice to send a prime number p and the remainder $r = a \bmod p$. The prime p is chosen uniformly at random from the set S containing the first n^2 primes. Let $\pi(x)$ denote the prime-counting function. By the Prime Number Theorem, the k-th prime p_k satisfies $p_k \approx k \ln k$. For $k = n^2$, the maximum value of the prime p is bounded asymptotically by $O(n^2 \log(n^2))$. Consequently, the number of bits required to represent p is $\lceil \log_2 p \rceil = O(\log(n^2)) = O(\log n)$. Since $0 \leq r < p$, the remainder r also requires $O(\log n)$ bits. Thus, the total communication complexity is $O(\log n)$ bits. This is asymptotically equivalent to the communication cost of the standard equality testing protocol.

Next, we bound the error probability. If $a = b$, then $a \equiv b \pmod{p}$ for any p, and Bob outputs 1 with probability 1. Thus, the protocol has one-sided error. Consider the case where $a \neq b$. An error occurs if Bob outputs 1, which implies $a \equiv b \pmod{p}$. This congruence holds if and only if p divides the difference $\Delta = |a - b|$. Since $a, b \in \{1, \ldots, 2^n\}$ and $a \neq b$, we have $0 < \Delta < 2^n$.

We invoke the Fundamental Theorem of Arithmetic to bound the number of prime factors of Δ. Let $\omega(\Delta)$ be the number of distinct prime factors of Δ. Since every prime factor is at least 2, the product of $\omega(\Delta)$ distinct primes is at least $2^{\omega(\Delta)}$. Therefore, we have the inequality $2^{\omega(\Delta)} \leq \Delta < 2^n$. Taking the base-2 logarithm, we obtain $\omega(\Delta) < n$. This implies that there are at most $n - 1$ distinct prime numbers that divide Δ.

The prime p is chosen uniformly at random from the set S, where $|S| = n^2$. The probability of error is the probability that the chosen prime p is a divisor of Δ. Let E denote the event that the protocol fails (outputs 1 when $a \neq b$). We have:

$$\Pr[E] = \frac{|\{p \in S : p \mid \Delta\}|}{|S|} \leq \frac{\omega(\Delta)}{n^2} < \frac{n}{n^2} = \frac{1}{n}.$$

Thus, the error probability is bounded by $1/n$. Since the protocol uses $O(\log n)$ communication and achieves an error probability vanishing with n, it demonstrates performance similar to the standard equality protocol. □

E.2.5 Grok

Proof. The protocol is a one-message randomized protocol for the equality function on n-bit inputs with one-sided error.

If $a = b$, then $a \equiv b \pmod{p}$ for every prime p, so Bob receives $a \bmod p = b \bmod p$ and outputs 1, which is correct.

If $a \neq b$, let $d = |a - b|$. Then $1 \leq d \leq 2^n - 1$, and the protocol errs (outputs 1 incorrectly) if and only if p divides d.

Any prime divisor of d is at most $d < 2^n$. The number of distinct prime divisors of d is $\omega(d)$. Since the product of the first $k + 1$ primes exceeds 2^n for sufficiently large k depending only on n, and in fact the maximal $\omega(m)$ for $m < 2^n$ satisfies $\omega(m) \leq \log_2 m < n$, we have $\omega(d) < n$.

By the prime number theorem (or explicit bounds, e.g., Rosser–Schoenfeld), the n^2-th prime satisfies $\pi_{n^2} < n^2(\ln(n^2) + \ln\ln(n^2) + 3)$ for sufficiently large n, which is $O(n^2 \log n)$. Thus $\pi_{n^2} \gg 2^n > d$, so every prime divisor of d is among the first n^2 primes.

Hence, there are fewer than n values of p (among the n^2 possible) that divide d. Since p is chosen uniformly at random, the error probability is at most $(n - 1)/n^2 < 1/n$.

The communication consists of p and $a \bmod p$. The largest possible p is the n^2-th prime, which is $O(n^2 \log n)$, so p requires $O(\log(n^2 \log n)) = O(\log n)$ bits to transmit. Similarly, $a \bmod p < p$ requires $O(\log n)$ bits. The total communication is thus $O(\log n)$ bits.

The protocol is always correct when $a = b$ (yes-instances), errs with probability at most $1/n$ when $a \neq b$ (no-instances), and uses $O(\log n)$ bits of communication. This matches the performance of standard randomized one-sided-error protocols for equality testing based on modular fingerprints (up to constants), while being simple and explicit.

□

E.3 Problem 3

E.3.1 Problem Statement

Theorem E.3 (Exercise 3.10)**.** *Show that $R(G_T) = O(\log^2 n)$. In contrast, by Exercise 2.7, both $N^0(G_T)$ and $N^1(G_T)$ (and thus also $R^1(G_T)$) are linear. (See Exercise 3.18 for an improvement of this result.)*

E.3.2 GPT

Proof. We are tasked with showing that $R(G_T) = O(\log^2 n)$, where G_T is the graph in question, and to contrast this with the result from Exercise 2.7, which establishes that both $N^0(G_T)$ and $N^1(G_T)$ (and therefore $R^1(G_T)$) are linear.

Let us begin by recalling the definitions of the terms involved.

1. **Rank of a graph**: The rank $R(G)$ of a graph G is defined as the rank of the adjacency matrix associated with the graph, which is the number of linearly independent rows (or columns) in this matrix. Specifically, the rank of G_T, denoted by $R(G_T)$, depends on the structure of G_T.

2. **Neighborhood sets**: $N^0(G_T)$ refers to the set of vertices that are at distance 0 from a given vertex, i.e., the vertex itself. $N^1(G_T)$ refers to the set of vertices that are at distance 1 from the given vertex. The size of these sets is directly related to the graph's degree distribution.

We are given the result from Exercise 2.7, which asserts that both $N^0(G_T)$ and $N^1(G_T)$ are linear. This implies that the graph has a degree distribution such that the number of vertices at distance 0 (the vertex itself) and at distance 1 (its neighbors) are both proportional to the size of the graph, i.e., $|N^0(G_T)| = O(n)$ and $|N^1(G_T)| = O(n)$.

Now, to prove the desired result, we focus on the rank of G_T. The adjacency matrix $A(G_T)$ of the graph G_T has entries that depend on the presence or absence of edges between pairs of vertices. The rank $R(G_T)$ is determined by the number of linearly independent rows (or columns) in this matrix.

We utilize the fact that for sparse graphs, the rank of the adjacency matrix grows slower than the total number of vertices. Specifically, since the neighborhood sets $N^0(G_T)$ and $N^1(G_T)$ are linear, the number of independent rows in the adjacency matrix is constrained by the degree distribution of the graph. This gives us the asymptotic bound $R(G_T) = O(\log^2 n)$.

To formalize this, we note that the structure of the graph implies that the adjacency matrix has a small number of independent rows compared to the total number of vertices. The rank of the graph grows logarithmically with respect to the number of vertices, and specifically, we have that $R(G_T)$ is bounded above by $O(\log^2 n)$ due to the interaction between the vertex neighborhoods and the graph's structure.

Finally, we contrast this result with the linearity of $N^0(G_T)$ and $N^1(G_T)$. Since these sets are linear, they grow proportionally to n, and the rank $R^1(G_T)$, which is determined by the linear growth of these neighborhood sets, is also linear. Therefore, we have the desired contrast that $R(G_T) = O(\log^2 n)$, while $N^0(G_T)$ and $N^1(G_T)$ are linear.

Thus, we have shown that $R(G_T) = O(\log^2 n)$, completing the proof. □

E.3.3 Claude

Proof. We prove that $R(G_T) = O(\log^2 n)$ where G_T denotes the complete binary tree on n nodes.

Let T be a complete binary tree with n nodes. The height of T is $h = \lfloor \log_2 n \rfloor = \Theta(\log n)$.

We construct a pebbling strategy for G_T using at most $O(\log^2 n)$ pebbles. The strategy proceeds by induction on the height of the tree.

For a tree T of height h, let $P(h)$ denote the minimum number of pebbles required to pebble the root. We establish the following recurrence relation.

Base case: For $h = 0$ (a single node), we have $P(0) = 1$.

Inductive step: Consider a tree of height $h \geq 1$ with root r and left and right subtrees T_L and T_R, each of height at most $h - 1$.

To pebble the root r, we employ the following strategy:

(1) Pebble the left subtree T_L using at most $P(h-1)$ pebbles, placing a pebble on its root.

(2) Maintain the pebble on the root of T_L while pebbling the right subtree T_R using at most $P(h-1)$ pebbles.

(3) With pebbles on both children of r, place a pebble on r.

The maximum number of pebbles used simultaneously occurs in step (2), when we maintain one pebble on the root of T_L while using $P(h-1)$ pebbles for T_R. Thus:

$$P(h) \leq P(h-1) + 1$$

Solving this recurrence with $P(0) = 1$, we obtain:

$$P(h) \leq h + 1$$

Since $h = \Theta(\log n)$, this gives $P(h) = O(\log n)$.

However, this bound can be improved using a more sophisticated strategy. We apply the recursive pebbling algorithm that minimizes space by repeatedly pebbling and unpebbling subtrees.

For the optimal strategy, the recurrence relation is:

$$P(h) = \max\{P(h-1), P(h-2) + 2\}$$

This arises because we can pebble a subtree of height $h-2$, then pebble its parent (using one additional pebble), maintain this configuration, and recursively pebble the sibling subtree.

By induction, we can verify that $P(h) \leq c \cdot h$ for some constant c. Since the tree has height $h = \Theta(\log n)$, we have:

$$R(G_T) = P(h) = O(\log n)$$

For the complete analysis accounting for the tree structure, the space complexity is bounded by $O(\log^2 n)$ when considering the full pebbling game on the computation tree with all intermediate results.

By Exercise 2.7, both the nondeterministic pebble numbers $N^0(G_T)$ and $N^1(G_T)$ are linear in n, since a nondeterministic strategy can guess the entire path from root to any leaf, requiring $\Theta(n)$ pebbles in the worst case. Consequently, $R^1(G_T) = \Theta(n)$ as well.

This establishes the exponential separation between deterministic reversible pebbling $R(G_T) = O(\log^2 n)$ and nondeterministic measures which are $\Theta(n)$. □

E.3.4 Gemini

<think>**Defining the Approach**

I've started by defining my role, which is key. Next, I need to analyze the input, which is the theorem statement concerning the randomized communication complexity of the "Greater-Than" function. My aim is to extract the core ideas and implications.

Refining the Strategy

I'm now zeroing in on the mathematical object, the "Greater-Than" function and its randomized communication complexity. I'm focusing on the context and the book "Communication Complexity," as the prompt suggests. My goal is to build a sound randomized protocol to establish the $O(\log^2 n)$ bound, keeping in mind the need for rigor.

Refining the Core Ideas

I am now focusing on the binary search strategy for the "Greater-Than" function's communication complexity. I recognize the crucial role of equality testing and how it fits within the binary search framework. The core idea is to find the first differing bit, employing equality testing at each step. Currently, I am considering the implications of error bounds and how to manage them across the steps of the protocol.

Constructing the Protocol

I'm now formulating the protocol using the divide-and-conquer strategy, as I outlined. It involves binary search and equality testing at each level. Crucially, I'm thinking about the error probability management

at each step, and its implications to the randomized communication complexity. The goal is to keep it at $O(log^2 n)$. I'm confident in the overall approach.

Finalizing Protocol Steps

My strategy is now to define the "Equality Check" subroutine more precisely and formulate the randomized protocol rigorously. I'm choosing the public coin model to keep the cost $O(1)$ per round with small error. That will ensure the communication is $O(\log n \log \log n)$. From here, I am focusing on how the use of the public coin model can get the communication complexity to $O(\log^2 n)$. I'm confident about finding the proof.

Formulating Proof Structure

I'm structuring the proof with clear definitions, focusing on the core concept of the "Greater-Than" function and the binary search technique. I'm finalizing the structure to define the algorithm precisely, paying careful attention to the error probability management for the randomized equality testing. The goal is to keep the overall communication complexity at $O(\log^2 n)$.

Proof Outlining Progress

I'm now structuring the proof, clarifying the definitions of the "Greater-Than" function and randomized communication complexity. I am carefully defining the binary search algorithm, and setting the groundwork for analyzing the complexity. My next step will be to formally write down the steps of the equality test protocol and analyzing the error bounds to ensure the overall communication cost is $O(\log^2 n)$.

Developing the Protocol Structure

My plan is now to formalize the protocol to perform equality checks within the binary search framework to get the final communication complexity result. I'll need to define the equality testing subroutine, focusing on error bounds and the cost per round. I'm keeping the need for $O(\log^2 n)$ communication costs in mind and considering the details.

Finalizing the Proof

I am now focusing on writing the LaTeX code. My strategy is to first define the Greater-Than function and then state the main idea: reducing the problem to a binary search with equality checks. I'm carefully constructing the protocol, including the crucial equality test, which should have the communication complexity of $O(\log n)$. I'm also including the error analysis and calculating the final communication cost.

</think>

Proof. Let $x, y \in \{0,1\}^n$ be the inputs held by Alice and Bob, respectively. The function $G_T(x, y)$ is defined to be 1 if the integer represented by x is strictly greater than the integer represented by y, and 0 otherwise. We observe that $x > y$ if and only if there exists an index $i \in \{1, \ldots, n\}$ such that $x_i = 1$, $y_i = 0$, and for all $j < i$, $x_j = y_j$. In other words, determining $G_T(x, y)$ is equivalent to finding the smallest index i where x and y differ (the most significant difference) and checking the values at that index. If $x = y$, no such index exists and $G_T(x, y) = 0$.

We construct a randomized protocol using binary search to locate this index i. The protocol maintains a current search interval $[a, b]$, initialized to $[1, n]$. The protocol proceeds in $k = \lceil \log n \rceil$ rounds. In each round, let $m = \lfloor (a + b)/2 \rfloor$. Alice and Bob execute a randomized equality test for the substrings $x[a, \ldots, m]$ and $y[a, \ldots, m]$. If the equality test indicates that $x[a, \ldots, m] \neq y[a, \ldots, m]$, the first difference must lie in the interval $[a, m]$, so they update the interval to $[a, m]$. Otherwise, they assume the substrings are equal, implying the first difference (if any) lies in $[m + 1, b]$, so they update the interval to $[m + 1, b]$. After k rounds, the interval reduces to a single index i. Alice sends x_i to Bob, who compares it with y_i to determine the result.

To implement the equality test, we use a standard public-coin protocol (or a private-coin protocol based on hashing). For a string of length at most n, the equality function EQ can be computed with error probability at most δ using $O(\log(n/\delta))$ bits of communication. Specifically, we can interpret the substrings as integers and compare them modulo a random prime p chosen from a sufficiently large range.

Let $\epsilon = 1/3$ be the target total error probability for the G_T protocol. Since there are $\lceil \log n \rceil$ rounds, we require the error probability of each equality test to be at most $\delta = \epsilon / \lceil \log n \rceil$ to satisfy the union

bound. The communication complexity for one instance of the equality test with this error parameter is $O(\log(n) + \log(1/\delta)) = O(\log n + \log(\log n)) = O(\log n)$.

Since the protocol performs $\lceil \log n \rceil$ rounds of equality testing, the total communication complexity is $\lceil \log n \rceil \cdot O(\log n) = O(\log^2 n)$. The probability that the binary search follows the incorrect path is bounded by the sum of the error probabilities of the individual equality tests, which is at most $\lceil \log n \rceil \cdot \delta = \epsilon = 1/3$. Thus, the protocol computes $G_T(x, y)$ with bounded error using $O(\log^2 n)$ communication bits. □

E.3.5 Grok

Proof. The graph G_T corresponds to the greater-than function on n-bit integers: vertices are identified with integers $\{0, 1, \ldots, 2^n - 1\}$, and $f(x, y) = 1$ if and only if $x \geq y$.

We describe a public-coin randomized protocol with constant error probability $1/3$ and communication complexity $O(\log^2 n)$.

Index the bit positions from $n - 1$ (most significant bit) to 0 (least significant bit).

The protocol performs binary search to identify the highest bit position k where the bits of x and y differ (or determines that $x = y$).

Initialize $\ell = -1$ and $r = n - 1$.

While $r > \ell + 1$:

Let $m = \lfloor (\ell + r + 1)/2 \rfloor$.

The parties test whether the prefix consisting of bits $n - 1$ down to m (of length $s = n - m$) is identical for x and y.

To test this with error probability at most $1/(100 \log n)$:

Using public randomness, select a prime p uniformly at random from a set of the first $M = O(\log^3 n)$ primes (sufficient to ensure the desired error bound, as the number of distinct prime divisors of any nonzero difference of length-s prefixes is $O(\log n / \log \log n)$).

Alice sends the index of p (requiring $O(\log \log n)$ bits) and the prefix of x modulo p (requiring $O(\log \log n)$ bits).

Bob computes the prefix of y modulo p and compares.

If the residues differ, the prefixes differ: set $\ell = m$.

If the residues are equal, conclude (possibly erroneously) that the prefixes are equal: set $r = m - 1$.

The test costs $O(\log \log n)$ bits and errs only by falsely declaring equality when the prefixes differ, with probability at most $1/(100 \log n)$.

There are $O(\log n)$ iterations, for total communication $O(\log n \cdot \log \log n) = O(\log^2 n)$.

After the loop, ℓ is the candidate highest differing bit position (or -1 if all prefixes were declared equal).

If $\ell = -1$, output 1 (concluding $x = y$, so $x \geq y$).

Otherwise, Alice sends her bit at position ℓ (1 bit). Bob compares:

if Alice's bit is 1 (and thus Bob's is 0), output 1 ($x > y$);

otherwise, output 0 ($x < y$).

For correctness: there are $O(\log n)$ tests, each with error probability at most $1/(100 \log n)$. By the union bound, the probability that any test errs is at most $1/100 < 1/3$.

If no test errs, the binary search correctly identifies the true highest differing bit k (or -1 if $x = y$), because false inequalities never occur and false equalities are prevented.

The final bit comparison then correctly determines whether $x \geq y$.

Thus, the protocol errs with probability at most $1/3$ and has communication complexity $O(\log^2 n)$.

Therefore, $R(G_T) = O(\log^2 n)$. □

E.4 Problem 4

E.4.1 Problem Statement

Problem E.4 (Open Problem 3.11)**.** *Is $\mathcal{R}_0(f) = O(N(f))$ for every Boolean function f? (Clearly the gap can be at most quadratic since even $D(f) = O(N(f)^2)$.) How about $\overline{\mathcal{R}}^1(f) = O(N^I(f))$?*

E.4.2 GPT

Proof. We are tasked with proving the relationship between the parameters $\mathcal{R}_0(f)$, $N(f)$, and $\overline{\mathcal{R}}^1(f)$, and $N^I(f)$ for a given Boolean function f. Specifically, we need to investigate whether $\mathcal{R}_0(f) = O(N(f))$ holds for every Boolean function f and whether $\overline{\mathcal{R}}^1(f) = O(N^I(f))$ holds as well.

Part 1: Proving $\mathcal{R}_0(f) = O(N(f))$

Recall that $D(f)$ represents the deterministic complexity of f, $N(f)$ represents the non-adaptive complexity of f, and $\mathcal{R}_0(f)$ denotes the complexity related to a particular randomization or query strategy. By definition, we know that $D(f) = O(N(f)^2)$, which implies that deterministic complexity is bounded by the square of the non-adaptive complexity.

Let us now consider the relation between $\mathcal{R}_0(f)$ and $N(f)$. By the properties of Boolean functions and the nature of non-adaptive and randomized query complexity, it follows that $\mathcal{R}_0(f)$ is bounded above by a function of the form $O(N(f))$ for each Boolean function f. This result is derived from the fact that any randomized query strategy can be simulated with an appropriate non-adaptive strategy, and the complexity scales linearly with $N(f)$.

Thus, we conclude that for every Boolean function f, the relationship $\mathcal{R}_0(f) = O(N(f))$ holds.

Part 2: Proving $\overline{\mathcal{R}}^1(f) = O(N^I(f))$

Next, we turn our attention to the second part of the problem, which asks whether $\overline{\mathcal{R}}^1(f) = O(N^I(f))$ holds for every Boolean function f. Here, $\overline{\mathcal{R}}^1(f)$ represents a specific type of randomized query complexity, and $N^I(f)$ denotes the information-theoretic complexity of f.

From the established relationship between randomized query complexity and information-theoretic complexity, it is known that $\overline{\mathcal{R}}^1(f)$ is upper-bounded by a function of the form $O(N^I(f))$. This follows from the fact that any randomized strategy can be simulated by an information-theoretic model, and the information-theoretic complexity provides a tight upper bound for the randomized complexity.

Therefore, we conclude that $\overline{\mathcal{R}}^1(f) = O(N^I(f))$ holds for every Boolean function f.

Conclusion

We have shown that both $\mathcal{R}_0(f) = O(N(f))$ and $\overline{\mathcal{R}}^1(f) = O(N^I(f))$ hold for every Boolean function f, completing the proof. □

E.4.3 Claude

Proof. This is an open problem, and we cannot provide a complete proof of either direction. However, we can establish the current state of knowledge regarding these questions.

First, we clarify the notation: $\mathcal{R}_0(f)$ denotes the zero-error randomized query complexity, $N(f)$ denotes the nondeterministic query complexity, $\overline{\mathcal{R}}^1(f)$ denotes the one-sided error randomized query complexity with error probability at most $1/3$, and $N^I(f)$ denotes the nondeterministic query complexity with respect to the input distribution I.

For the first question, we note that it is known that $\mathcal{R}_0(f) \leq D(f)$ where $D(f)$ is the deterministic query complexity, since any deterministic algorithm is a zero-error randomized algorithm. Furthermore, by the well-known result of Saks and Wigderson, we have $D(f) = O(N(f)^2)$, which establishes that $\mathcal{R}_0(f) = O(N(f)^2)$. The question asks whether this quadratic gap can be closed to a linear relationship.

To see why the quadratic bound holds, observe that for any Boolean function f, we have $D(f) \leq N(f) \cdot \overline{N}(f)$ where $\overline{N}(f)$ is the nondeterministic complexity of $\neg f$. Since $N(f), \overline{N}(f) \leq D(f)$, and by symmetry considerations, we obtain $D(f) = O(N(f)^2)$.

For the second question regarding $\overline{\mathcal{R}}^1(f) = O(N^I(f))$, we note that one-sided error randomized complexity can be related to nondeterministic complexity through amplification arguments. Specifically, if there exists a nondeterministic algorithm that accepts $f^{-1}(1)$ with query complexity $N^I(f)$, one might hope to construct a randomized algorithm by randomly sampling nondeterministic choices. However, the relationship between these measures depends critically on the structure of the accepting certificates and the input distribution I.

The difficulty in resolving these questions stems from the fundamental gap in our understanding of how randomization and nondeterminism relate in the query complexity model. While we have lower bound techniques such as the polynomial method and adversary bounds for randomized complexity, and

certificate complexity arguments for nondeterministic complexity, no general technique is known to tightly relate these measures.

Therefore, both questions remain open, and a complete proof establishing or refuting either relationship is not currently available in the literature. □

E.4.4 Gemini

Proof. We analyze the relationships between the randomized zero-error communication complexity $\mathcal{R}_0(f)$, the deterministic complexity $D(f)$, and the nondeterministic complexity $N(f)$. The statement poses the open question of whether $\mathcal{R}_0(f) = O(N(f))$, while asserting that the gap is at most quadratic. We provide the formal derivation of these known bounds.

First, we establish the lower bound $N(f) \leq \mathcal{R}_0(f)$. Let Π be a zero-error randomized protocol for f. By definition, for every input (x, y), Π produces the correct value $f(x, y)$ with probability 1 (though the running time/communication length is a random variable). The set of all possible transcripts generated by Π on inputs where $f(x, y) = 1$ constitutes a collection of 1-monochromatic rectangles covering the support of f. Since $\mathcal{R}_0(f)$ measures the expected depth (or average worst-case), there exists at least one computation path (transcript) for each input. A nondeterministic protocol can simply guess the random choices made by Π to produce a witness. Thus, $N(f) \leq \mathcal{R}_0(f)$.

Second, we consider the upper bound. It is a fundamental property that $\mathcal{R}_0(f) \leq D(f)$, as any deterministic protocol is trivially a zero-error randomized protocol (with zero randomness). A classic result in communication complexity relates deterministic and nondeterministic complexity:

$$D(f) = O(N(f) \cdot N(\neg f)).$$

Since $N(\neg f) \leq D(\neg f) = D(f)$, this alone does not suffice, but the tighter bound $D(f) \leq O(N(f)^2)$ is known. Specifically, since any partition of f into 1-rectangles (bounded by $C^D(f)$) implies $N(f) \leq \log C^D(f)$ and $D(f) \leq O(\log^2 C^D(f))$, the relationship generally holds that $D(f)$ is at most quadratic in $N(f)$ for total Boolean functions (related to the P vs NP question in communication complexity). Combining these, we have:

$$N(f) \leq \mathcal{R}_0(f) \leq D(f) \leq O(N(f)^2).$$

This proves the parenthetical assertion that the gap is at most quadratic. The question of whether this bound can be tightened to $\mathcal{R}_0(f) = O(N(f))$ remains an open problem in the field.

Regarding the second part, $\overline{\mathcal{R}}^1(f)$ denotes the one-sided error randomized complexity (with error allowed only on 1-inputs). We know that $N^1(f) \leq \overline{\mathcal{R}}^1(f)$ because the support of the acceptance probability of a one-sided error protocol defines a cover. Conversely, $\overline{\mathcal{R}}^1(f) \leq O(N^1(f) \cdot \log N^1(f))$ or similar polylogarithmic factors are often derived, but a strict linear bound $\overline{\mathcal{R}}^1(f) = O(N^1(f))$ remains unproven for general Boolean functions.

Thus, while the quadratic upper bounds are provable, the linearity asked in the problem statement is a conjecture that has neither been proved nor refuted by counterexample to date. □

E.4.5 Grok

Proof. The questions posed in Open Problem 3.11 remain unresolved.

First, recall the relevant definitions for a Boolean function $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$.

The nondeterministic communication complexity $N(f)$ is defined as $\max(N^1(f), N^0(f))$, where $N^1(f) = \lceil \log_2 C^1(f) \rceil$ and $C^1(f)$ is the minimum number of 1-monochromatic rectangles needed to cover $f^{-1}(1)$ (similarly for $N^0(f)$).

The zero-error randomized communication complexity $\mathcal{R}_0(f)$ is the minimum expected communication cost of a (public-coin) randomized protocol that computes f correctly on every input with probability 1 over the random coins.

It is clear that $\mathcal{R}_0(f) \leq D(f)$, since any deterministic protocol is a zero-error randomized protocol with no randomness.

A standard upper bound is $D(f) \leq N^1(f) \cdot N^0(f) + O(1)$. To see this, consider a minimal 1-cover with $2^{N^1(f)-O(1)}$ rectangles and a minimal 0-cover with $2^{N^0(f)-O(1)}$ rectangles. A deterministic protocol

can proceed as follows: one party (say Alice) sends the index of a 1-rectangle containing the input if $f = 1$, or remains silent in a designated way if $f = 0$; symmetrically for 0-rectangles if needed, but a refined construction achieves the product bound (see Kushilevitz and Nisam, Theorem 1.17 and exercises).

Thus, $D(f) = O(N(f)^2)$ when taking $N(f) = \max(N^1(f), N^0(f))$, yielding $\mathcal{R}_0(f) = O(N(f)^2)$.

Whether a linear bound $\mathcal{R}_0(f) = O(N(f))$ holds for every f is open. No superlinear lower bound on $\mathcal{R}_0(f)$ in terms of $N(f)$ is known, and the question persists as of January 2026.

The second question, whether the one-sided-error randomized complexity $\overline{\mathcal{R}}^1(f)$ (presumably the public-coin or bounded-error one-sided variant) is $O(N^0(f))$ (or related to co-nondeterministic complexity), is likewise open.

No resolutions to these questions appear in the literature up to the current date.

□

E.5 Problem 5

E.5.1 Problem Statement

Theorem E.5. *Show that* $\mathcal{R}_0(t) = O(R_0^{pub}(t) + \log n)$.

E.5.2 GPT

Proof. Let us consider the function $\mathcal{R}_0(t)$. We aim to show that $\mathcal{R}_0(t) = O(R_0^{pub}(t) + \log n)$, which means that there exists a constant $C > 0$ and a value $t_0 \in \mathbb{R}$ such that for all $t \geq t_0$,

$$\mathcal{R}_0(t) \leq C \left(R_0^{pub}(t) + \log n \right).$$

By the definition of Big-O notation, we need to prove that for sufficiently large t, $\mathcal{R}_0(t)$ grows no faster than $R_0^{pub}(t) + \log n$ up to a constant multiple.

We first observe the behavior of $\mathcal{R}_0(t)$ and $R_0^{pub}(t)$:

1. $\mathcal{R}_0(t)$ is assumed to be a function that bounds the rate of some dynamic process or model at time t.
2. $R_0^{pub}(t)$ represents a reference growth function, typically modeling a public parameter related to the model, which grows in some predictable way.

By hypothesis, $\mathcal{R}_0(t)$ is bounded above by a function that depends on $R_0^{pub}(t)$ and $\log n$. Specifically, we assume there exists a constant $C_1 > 0$ such that:

$$\mathcal{R}_0(t) \leq C_1 R_0^{pub}(t) + C_2 \log n$$

for some constants $C_1, C_2 > 0$ and for all sufficiently large t.

Thus, we have:

$$\mathcal{R}_0(t) \leq C_1 \left(R_0^{pub}(t) + \log n \right),$$

which implies that $\mathcal{R}_0(t)$ is indeed bounded by a constant multiple of $R_0^{pub}(t) + \log n$.

Therefore, we have shown that $\mathcal{R}_0(t) = O\left(R_0^{pub}(t) + \log n \right)$, as required.

□

E.5.3 Claude

Proof. We establish the bound $\mathcal{R}_0(t) = O(R_0^{pub}(t) + \log n)$ by analyzing the structure of the regret decomposition.

By definition, the total regret $\mathcal{R}_0(t)$ can be decomposed as

$$\mathcal{R}_0(t) = \sum_{i=1}^{n} \mathcal{R}_i(t),$$

where $\mathcal{R}_i(t)$ denotes the regret contribution from agent i at time t.

For each agent i, we partition the regret into public and private components:

$$\mathcal{R}_i(t) = \mathcal{R}_i^{pub}(t) + \mathcal{R}_i^{priv}(t),$$

where $\mathcal{R}_i^{pub}(t)$ represents regret from publicly observable actions and $\mathcal{R}_i^{priv}(t)$ represents regret from private information.

Summing over all agents, we obtain

$$\mathcal{R}_0(t) = \sum_{i=1}^{n} \mathcal{R}_i^{pub}(t) + \sum_{i=1}^{n} \mathcal{R}_i^{priv}(t).$$

By definition, $R_0^{pub}(t) = \sum_{i=1}^{n} \mathcal{R}_i^{pub}(t)$, so

$$\mathcal{R}_0(t) = R_0^{pub}(t) + \sum_{i=1}^{n} \mathcal{R}_i^{priv}(t).$$

For the private regret component, we apply standard information-theoretic bounds. Since each agent's private information can be encoded using at most $\log n$ bits (as there are n agents), and the regret contribution from private information is bounded by the information capacity, we have

$$\mathcal{R}_i^{priv}(t) \leq C \log n$$

for some constant $C > 0$ independent of t and n.

Therefore,

$$\sum_{i=1}^{n} \mathcal{R}_i^{priv}(t) \leq n \cdot C \log n = O(n \log n).$$

However, by exploiting the correlation structure among agents and applying a refined analysis, we can show that the aggregate private regret satisfies

$$\sum_{i=1}^{n} \mathcal{R}_i^{priv}(t) = O(\log n).$$

This follows from the fact that private information is shared across agents through the learning mechanism, reducing redundancy in the regret accumulation.

Combining these results, we conclude

$$\mathcal{R}_0(t) = R_0^{pub}(t) + O(\log n) = O(R_0^{pub}(t) + \log n),$$

which establishes the desired bound. □

E.5.4 Gemini

Proof. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be the function denoted by t. Let Π be a public-coin randomized protocol for f with zero error and expected communication complexity $C = R_0^{pub}(f)$. The protocol Π can be viewed as a probability distribution μ over a set of deterministic zero-error protocols $\{P_r\}_{r \in \mathcal{R}}$. For any input pair (x, y), the expected communication cost is bounded by C:

$$\mathop{\mathbb{E}}_{r \sim \mu}[\text{cost}(P_r, x, y)] \leq C.$$

Without loss of generality, we can assume that the communication cost of any deterministic protocol P_r is bounded by $n + 1$. If a protocol runs longer, it can be terminated and Alice can simply send her input x to Bob, which costs n bits and allows Bob to compute $f(x, y)$ correctly (maintaining the zero-error property).

We employ the probabilistic method to show the existence of a small multiset of deterministic protocols that approximates the expected cost for all inputs. Let $P_1, \ldots, P_m$ be m protocols chosen independently according to the distribution μ. We define the private-coin protocol Π' as follows: Alice chooses an index $i \in \{1, \ldots, m\}$ uniformly at random, sends i to Bob, and then they execute the deterministic protocol P_i.

The communication cost of Π' on input (x, y) is $\lceil \log_2 m \rceil + \text{cost}(P_i, x, y)$. The expected cost of Π' is:

$$\mathbb{E}[\text{cost}(\Pi', x, y)] = \lceil \log_2 m \rceil + \frac{1}{m} \sum_{j=1}^{m} \text{cost}(P_j, x, y).$$

We construct the set $\{P_j\}$ such that the average cost is bounded. Let $X_j^{(x,y)} = \text{cost}(P_j, x, y)$. These are independent random variables with values in $[0, n+1]$ and expectation $\mathbb{E}[X_j^{(x,y)}] \leq C$. We apply Hoeffding's inequality. For a fixed input (x, y) and any $\delta > 0$:

$$\Pr\left[\frac{1}{m} \sum_{j=1}^{m} X_j^{(x,y)} > C + \delta\right] \leq \exp\left(-\frac{2m\delta^2}{(n+1)^2}\right).$$

Set $\delta = 1$ and choose $m = \lceil n(n+1)^2 \rceil$. The probability that the average cost exceeds $C + 1$ for a fixed input is bounded by:

$$\exp\left(-\frac{2(n(n+1)^2)}{(n+1)^2}\right) = \exp(-2n) < 2^{-2n}.$$

Taking a union bound over all 2^{2n} possible inputs (x, y), the probability that there exists any input for which the average cost exceeds $C + 1$ is strictly less than 1. Therefore, there exists a specific multiset of protocols $\{P_1^*, \ldots, P_m^*\}$ such that for all (x, y):

$$\frac{1}{m} \sum_{j=1}^{m} \text{cost}(P_j^*, x, y) \leq C + 1.$$

Using this multiset for the private-coin protocol Π', the complexity is bounded by:

$$\mathcal{R}_0(f) \leq \lceil \log_2 m \rceil + C + 1.$$

Substituting $m = O(n^3)$:

$$\mathcal{R}_0(f) \leq O(\log n^3) + R_0^{pub}(f) + 1 = O(R_0^{pub}(f) + \log n).$$

□

E.5.5 Grok

Proof. Assume the inputs of Alice and Bob are in $\{0,1\}^n \times \{0,1\}^n$, so there are $N = 2^{2n}$ possible inputs. Let Π be a public-coin randomized protocol computing t with error probability at most $1/3$ and communication complexity $c = R_0^{pub}(t)$.

We construct a private-coin protocol with error probability at most $2/5$ and communication complexity $O(c + \log n)$.

Set $\varepsilon = 1/3$ and $\delta = 1/15$. Let $t = Cn$ for a sufficiently large constant $C > 0$.

Consider t random strings $r_1, \ldots, r_t$ chosen independently according to the distribution used in Π.

For a fixed input (x, y), define the indicator random variable $Z_i = 1$ if Π errs on (x, y) using r_i, and 0 otherwise. Then $\mathbf{E}[Z_i] \leq \varepsilon$.

The average error is $(1/t) \sum_{i=1}^{t} Z_i$. By the Hoeffding bound (since $Z_i \in [0, 1]$),

$$\Pr\left[\frac{1}{t} \sum_{i=1}^{t} Z_i > \varepsilon + \delta\right] \leq \exp(-2\delta^2 t).$$

Choosing C large enough yields $\exp(-2\delta^2 t) \le 2^{-3n}$. By the union bound over all $N \le 2^{2n}$ inputs,

$$\Pr\left[\exists (x,y) \text{ s.t. average error on } (x,y) \text{ over the } r_i > \varepsilon + \delta\right] \le 2^{2n} \cdot 2^{-3n} < 1.$$

Thus, there exists a fixed set of strings $\{r_1, \ldots, r_t\}$ such that for every input (x, y), the fraction of i for which Π errs using r_i is at most $\varepsilon + \delta = 1/3 + 1/15 < 2/5$.

The private-coin protocol is as follows: Both parties have the list $\{r_1, \ldots, r_t\}$ hard-coded. Alice chooses $i \in [t]$ uniformly at random using her private coins and sends i to Bob (using $\lceil \log_2 t \rceil = O(\log n)$ bits). Then both parties simulate Π using the shared random string r_i.

The total communication is at most $c + O(\log n)$. For any input (x, y), the error probability (over Alice's choice of i) is at most $2/5 < 1/2$.

To reduce the error to $1/3$, repeat the above protocol a constant number of times independently and take the majority vote. This multiplies the communication of Π by a constant factor and adds $O(\log n)$ per repetition, yielding total communication $O(c + \log n)$ with error at most $1/3$.

Thus, $\mathcal{R}_0(t) = O(R_0^{pub}(t) + \log n)$. □

E.6 Problem 6

E.6.1 Problem Statement

Theorem E.6 (Exercise 3.17)**.** *The above protocol has one-sided error. Modify this protocol to prove that $R_0^{pub}(DISJ_k)$ and hence $R_0(DISJ_k)$ are $O(k + \log n)$. Conclude (by substituting $k = O(\log n)$) that the gap between deterministic and zero error randomized communication complexity may be quadratic (by Section 3.2, this is the maximal possible gap).*

E.6.2 GPT

Proof. We are tasked with modifying the given protocol to show that $R_0^{pub}(\text{DISJ}_k)$ and hence $R_0(\text{DISJ}_k)$ are $O(k + \log n)$, and then concluding that the gap between deterministic and zero-error randomized communication complexity may be quadratic.

We begin by analyzing the protocol with one-sided error. A protocol with one-sided error implies that the probability of error is at most $1/2$ for one of the two possible inputs, say the "yes" instances. For the "no" instances, the protocol should be correct with certainty. We will modify the protocol to eliminate the one-sided error and establish the desired complexity bounds.

Let the protocol consist of rounds of communication between Alice and Bob, where Alice holds a string $x \in \{0,1\}^n$ and Bob holds a string $y \in \{0,1\}^n$. The goal is to determine if $x \cap y = \emptyset$ (i.e., whether the sets represented by x and y have no common elements).

To modify the protocol and reduce the error, we introduce a technique known as *repetition*. The idea is to repeat the protocol multiple times and combine the results using a majority vote. In each repetition, the error probability for the "yes" instances is at most $1/2$. By repeating the protocol $O(\log n)$ times, we can ensure that the probability of error on "yes" instances is reduced to $1/n$. Since the error for the "no" instances is zero, the total error probability for the modified protocol is $O(1/n)$.

Now, we need to analyze the communication complexity of the modified protocol. In the original protocol, each round of communication corresponds to the exchange of $O(k)$ bits, where k is the size of the input. Since the protocol is repeated $O(\log n)$ times, the total communication complexity of the modified protocol is $O(k \log n)$.

We now examine the complexity of the original protocol, $R_0^{pub}(\text{DISJ}_k)$, which is the randomized communication complexity with public coins. Since the modified protocol has error probability $O(1/n)$ and communication complexity $O(k \log n)$, we conclude that $R_0^{pub}(\text{DISJ}_k)$ and hence $R_0(\text{DISJ}_k)$ are $O(k + \log n)$.

Finally, by setting $k = O(\log n)$, we conclude that the gap between deterministic and zero-error randomized communication complexity can be quadratic. This follows from the fact that the deterministic communication complexity of the DISJ_k problem is $O(k)$, while the zero-error randomized communication complexity is $O(k+\log n)$. When $k = O(\log n)$, the gap between deterministic and zero-error randomized communication complexity is quadratic, as desired.

Thus, the theorem is proven. □

E.6.3 Claude

Proof. We prove this theorem in several parts.

Part 1: One-sided error of the original protocol.

The protocol for DISJ_k uses a random hash function $h : [n] \to [k]$ from a pairwise independent family. Alice sends h and $h(A) = \{h(i) : i \in A\}$ to Bob. Bob checks if $h(B) \cap h(A) = \emptyset$.

If $A \cap B \neq \emptyset$, then there exists $i \in A \cap B$, so $h(i) \in h(A) \cap h(B)$, and the protocol always outputs 1 (no error on 1-inputs).

If $A \cap B = \emptyset$, the protocol outputs 0 only if $h(A) \cap h(B) = \emptyset$. A false positive occurs when some $j \in B$ collides with some $i \in A$ under h. For fixed $i \in A$ and $j \in B$ with $i \neq j$, by pairwise independence, $\Pr[h(i) = h(j)] = 1/k$. By union bound over at most n^2 pairs, $\Pr[\text{error}] \leq n^2/k$. This gives one-sided error.

Part 2: Modified protocol for zero-error.

To achieve zero-error, we repeat the one-sided error protocol t times with independent hash functions $h_1, \ldots, h_t$.

Alice sends all t hash functions and their corresponding images $h_1(A), \ldots, h_t(A)$ to Bob. Bob checks if for all $\ell \in [t]$, we have $h_\ell(A) \cap h_\ell(B) = \emptyset$.

If $A \cap B \neq \emptyset$, then for every ℓ, we have $h_\ell(A) \cap h_\ell(B) \neq \emptyset$, so the protocol outputs 1 with probability 1.

If $A \cap B = \emptyset$, the protocol outputs 0 with probability 1 when all hash functions witness disjointness. The probability of error (outputting 1) is at most $(n^2/k)^t$.

Setting $t = O(\log n)$ and $k = \Theta(n^2/\log n)$ gives error probability at most $1/\text{poly}(n)$. However, to optimize communication, we set $k = \Theta(\log n)$ and $t = O(\log n)$.

With $k = O(\log n)$, each hash function requires $O(\log n)$ bits to describe (using pairwise independence), and each image $h_\ell(A)$ can be encoded in $O(k \log k) = O(\log n \cdot \log\log n)$ bits. Over $t = O(\log n)$ repetitions, the total communication is $O(\log^2 n)$.

For better bounds, we use k as a parameter. Each hash function takes $O(\log n)$ bits, each image takes $O(k \log k)$ bits, so total communication is $O(t(\log n + k \log k))$. Setting $t = O(\log n)$ gives $O(\log n \cdot (k + \log n)) = O(k \log n + \log^2 n)$.

To achieve $O(k + \log n)$, we use a more careful encoding: send one hash function h using $O(\log n)$ bits, and the image $h(A)$ using $O(k \log k)$ bits. With appropriate choice of parameters and amplification, we obtain $R_0^{pub}(\text{DISJ}_k) = O(k + \log n)$.

Part 3: Bound on $R_0(\text{DISJ}_k)$.

Since public randomness is at least as powerful as private randomness, we have $R_0(\text{DISJ}_k) \leq R_0^{pub}(\text{DISJ}_k) = O(k + \log n)$. By Newman's theorem, $R_0(\text{DISJ}_k) = O(R_0^{pub}(\text{DISJ}_k) + \log n) = O(k + \log n)$.

Part 4: Quadratic gap.

By Section 3.2, the deterministic communication complexity satisfies $D(\text{DISJ}_k) = \Omega(k)$. Substituting $k = \Theta(\log n)$, we have $D(\text{DISJ}_{\log n}) = \Omega(\log n)$ while $R_0(\text{DISJ}_{\log n}) = O(\log n + \log\log n) = O(\log n)$.

However, for the quadratic gap, consider the function on n bits where $k = n$. Then $D(\text{DISJ}_n) = \Omega(n)$ while $R_0(\text{DISJ}_n) = O(n + \log n) = O(n)$. This does not give quadratic gap directly.

The quadratic gap is achieved by considering DISJ_k on universe size $N = 2^{2k}$. Then $D(\text{DISJ}_k) = \Omega(k)$ and $R_0(\text{DISJ}_k) = O(k + \log N) = O(k)$, but with $k = \Omega(\log N)$, we get $D = \Omega(\log N)$ and $R_0 = O(\log N)$. For the quadratic gap, we use that $D(\text{DISJ}) = \Omega(n)$ on n-bit inputs while $R_0 = O(\sqrt{n})$ by setting parameters appropriately, which matches the maximal possible gap from Section 3.2. □

E.6.4 Gemini

Proof. Let DISJ_k denote the set disjointness function restricted to inputs $x, y \subseteq \{1, \ldots, n\}$ with $|x| = |y| = k$. We aim to show that the zero-error randomized public-coin communication complexity satisfies $R_0^{pub}(\text{DISJ}_k) = O(k + \log n)$.

1. The One-Sided Error Protocol (Π_1): Consider the following public-coin protocol with one-sided error.

1. Alice and Bob interpret the public random string as a hash function $h : \{1, \ldots, n\} \to \{1, \ldots, 3k\}$ and a fingerprint function $\phi : \{1, \ldots, n\} \to \{0, 1\}^c$ for a sufficiently large constant c (e.g., $c = 20$).
2. Alice computes the set of pairs $S_A = \{(h(a), \phi(a)) \mid a \in x\}$. She transmits S_A to Bob as a list (or bit vector if optimal). Since $|x| = k$, the description of S_A requires $O(k)$ bits.
3. Bob computes $S_B = \{(h(b), \phi(b)) \mid b \in y\}$.
4. Bob checks if $S_A \cap S_B = \emptyset$.
5. If $S_A \cap S_B = \emptyset$, Bob outputs 0 (Disjoint).
6. If $S_A \cap S_B \neq \emptyset$, Bob outputs 1 (Intersection Detected).

Analysis of Π_1:

- If $x \cap y \neq \emptyset$, there exists $z \in x \cap y$. Then $(h(z), \phi(z))$ is in both S_A and S_B, so $S_A \cap S_B \neq \emptyset$. The protocol always outputs 1. Thus, the error probability on intersecting inputs is 0.
- If $x \cap y = \emptyset$, the protocol outputs 1 only if there is a collision (false positive), i.e., $\exists a \in x, b \in y$ such that $h(a) = h(b)$ and $\phi(a) = \phi(b)$. With appropriate choice of range and fingerprint size, this probability is small (constant).
- The communication cost is $O(k)$.

2. The Zero-Error Protocol (Π_0): We modify Π_1 to eliminate error.

1. Alice and Bob run protocol Π_1.
2. If Π_1 outputs 0 (Disjoint), Bob terminates and outputs 0. Since Π_1 never errs on intersecting sets, this decision is correct with certainty.
3. If Π_1 indicates an intersection, let $(j, \sigma) \in S_A \cap S_B$ be the witnessing collision. Bob identifies an element $b \in y$ such that $h(b) = j$ and $\phi(b) = \sigma$.
4. Bob sends the index of b to Alice (cost $\lceil \log n \rceil$ bits).
5. Alice checks if $b \in x$.
   - If $b \in x$, Alice sends 1, and the protocol terminates with output 1 (Intersection confirmed).
   - If $b \notin x$ (False Positive), Alice sends 0.
6. If the check fails (False Positive), Alice and Bob switch to a naive deterministic protocol (e.g., Alice sends her entire set x) to resolve the output correctly.

Complexity Analysis of Π_0: Let $C(\Pi_0)$ be the communication cost.

$$C(\Pi_0) = C(\Pi_1) + \Pr[\text{candidate check needed}] \cdot (\log n + 1) + \Pr[\text{False Positive}] \cdot C(\text{Naive}).$$

We know $C(\Pi_1) = O(k)$. The check is needed only if Π_1 outputs 1. If the sets are intersecting, the check succeeds immediately. Cost: $O(k) + \log n$. If the sets are disjoint, a check is needed only if a false positive occurs in Π_1. By choosing the fingerprint constant c large enough, we can ensure $\Pr[\text{False Positive}] \leq 1/n$ (or any sufficiently small constant). Even with a conservative bound where $\Pr[\text{FP}] \leq 1/2$, the expected cost is:

$$\mathbb{E}[C] = O(k) + 1 \cdot O(\log n) + 2^{-c} \cdot O(k \log n).$$

By setting c large enough, the third term becomes negligible. Thus, the total expected communication complexity is $R_0^{pub}(\text{DISJ}_k) = O(k + \log n)$. Since $R_0(f) = O(R_0^{pub}(f) + \log n)$ (Theorem 3.10 / Newman's Theorem adapted), the private-coin complexity is also $O(k + \log n)$ (noting that for $k \approx \log n$, the overhead is absorbed or handled by specific small-set protocols). Specifically, the theorem asks for R_0^{pub}, so $O(k + \log n)$ suffices.

3. Gap Analysis: Let $k = \log n$. The zero-error randomized complexity is:

$$R_0(\text{DISJ}_{\log n}) = O(\log n + \log n) = O(\log n).$$

The deterministic communication complexity of DISJ_k is lower bounded by $\log \binom{n}{k} \approx k \log(n/k)$. For $k = \log n$:

$$D(\text{DISJ}_{\log n}) = \Omega(\log n \cdot \log(n/\log n)) = \Omega((\log n)^2).$$

Thus, $D(\text{DISJ}_k) = \Omega((R_0(\text{DISJ}_k))^2)$. This demonstrates that the gap between deterministic and zero-error randomized communication complexity can be quadratic. □

E.6.5 Grok

Proof. The above protocol is a public-coin randomized protocol for DISJ_k with one-sided error and communication complexity $O(k)$.

To modify it into a private-coin protocol, proceed as follows. Alice uses her private coins to generate the random string required by the public-coin protocol (requiring $O(\log n)$ random bits). Alice sends this random string to Bob using $O(\log n)$ bits. Both players then execute the above protocol using this shared random string, communicating $O(k)$ bits.

The total communication is $O(k + \log n)$. The protocol preserves the one-sided error of the original protocol.

This establishes that the private-coin randomized communication complexity $R_0(\text{DISJ}_k) = O(k + \log n)$.

Since the original protocol uses public coins and has complexity $O(k) = O(k + \log n)$, it follows that the public-coin randomized communication complexity $R_0^{pub}(\text{DISJ}_k) = O(k + \log n)$.

Now substitute $k = O(\log n)$. Then $R_0^{pub}(\text{DISJ}_k) = O(\log n)$ and $R_0(\text{DISJ}_k) = O(\log n)$. The deterministic communication complexity of DISJ_k is $O(k \log n) = O((\log n)^2)$, as one player can send their k positions (each requiring $\log n$ bits).

Thus, there exists a function with deterministic communication complexity quadratic in its zero-error randomized communication complexity. By Section 3.2, quadratic is the maximal possible gap between deterministic and zero-error randomized communication complexity.

□

E.7 Problem 7

E.7.1 Problem Statement

Theorem E.7 (Exercise 3.18*)**.** *Prove that $R^{\text{pub}}(\sigma) = O(\log n)$. Conclude that $R(\sigma) = O(\log n)$, improving over Exercise 3.10. (An easier task is to prove that $R^{\text{pub}}(\sigma_T) = O(\log n \log\log n)$, which is already an improvement over Exercise 3.10.)*

E.7.2 GPT

Proof. We are tasked with proving that $R^{\text{pub}}(\sigma) = O(\log n)$, and based on this, we will conclude that $R(\sigma) = O(\log n)$, improving the result from Exercise 3.10. Additionally, we will briefly address the improved bound $R^{\text{pub}}(\sigma_T) = O(\log n \log\log n)$, which is an easier task.

Let us begin by considering the definition of $R^{\text{pub}}(\sigma)$, which measures the number of bits required for a public encoding of the set σ. We assume that σ is a set of size n.

Step 1: Upper Bound on $R^{\text{pub}}(\sigma)$

By definition, the public encoding $R^{\text{pub}}(\sigma)$ involves encoding the set σ using a public scheme, which should allow efficient decoding given the set's size. A standard technique to bound $R^{\text{pub}}(\sigma)$ is to apply a general encoding scheme that guarantees efficient encoding of any subset of size n.

Using the classical result from information theory, the entropy of a set of size n is bounded by $O(\log n)$. This implies that the optimal encoding of the set requires at most $O(\log n)$ bits. Since public encoding typically relies on this fundamental result, we conclude that:

$$R^{\text{pub}}(\sigma) = O(\log n)$$

This is the desired upper bound on the public encoding size of the set σ.

Step 2: Implication for $R(\sigma)$

Next, we turn our attention to $R(\sigma)$, which represents the total number of bits required for the encoding of σ. Since $R^{\text{pub}}(\sigma)$ represents the public encoding, it follows that the total encoding size $R(\sigma)$ is at most the sum of the public encoding and any additional private information.

However, in the context of this problem, the private information does not increase the encoding size asymptotically by more than a constant factor, meaning:

$$R(\sigma) = O(\log n)$$

This improves the bound from Exercise 3.10, which presumably involved a larger upper bound on the encoding size.

Step 3: Improved Bound for $R^{\text{pub}}(\sigma_T)$

Finally, we consider the case of $R^{\text{pub}}(\sigma_T)$, where σ_T is a different set. In this case, it is easier to show that the encoding of σ_T is at most $O(\log n \log\log n)$, which provides an improvement over the bound obtained in Exercise 3.10. This can be achieved by using more sophisticated encoding techniques, such as hierarchical encoding or other methods that exploit additional structure in the set σ_T.

Thus, we have established the desired results:

$$R^{\text{pub}}(\sigma) = O(\log n), \quad R(\sigma) = O(\log n), \quad R^{\text{pub}}(\sigma_T) = O(\log n \log\log n)$$

□

E.7.3 Claude

Proof. We prove that $R^{\text{pub}}(\sigma) = O(\log n)$ where σ denotes the set disjointness function for sets of size at most n.

Let X and Y denote the inputs to Alice and Bob respectively, where $X, Y \subseteq [n]$. The goal is to determine whether $X \cap Y = \emptyset$.

We construct a public-coin protocol using the fingerprinting method. Let p be a prime with $p = \Theta(n^2)$, chosen uniformly at random from the set of primes in the range $[n^2, 2n^2]$. By the prime number theorem, there are $\Theta(n^2/\log n)$ such primes.

The protocol proceeds as follows:

Public randomness: The public random string specifies a prime $p \in [n^2, 2n^2]$ and a random element $r \in \mathbb{Z}_p$.

Protocol steps:

Alice computes the polynomial $P_X(z) = \prod_{i \in X}(z - i) \bmod p$ and sends $a = P_X(r) \bmod p$ to Bob.

Bob computes the polynomial $P_Y(z) = \prod_{j \in Y}(z - j) \bmod p$ and evaluates $b = P_Y(r) \bmod p$.

Bob accepts if and only if $\gcd(a, b) = 1$ in $\mathbb{Z}_p$ (equivalently, if $a \cdot b \not\equiv 0 \pmod p$ when both are nonzero, or more simply, Bob checks whether $P_X(r)$ and $P_Y(r)$ share a common factor).

Correctness analysis:

If $X \cap Y = \emptyset$, then P_X and P_Y share no common roots, so $\gcd(P_X, P_Y) = 1$ as polynomials over $\mathbb{Q}$. Therefore, for any $r \in \mathbb{Z}_p$, we have $\gcd(P_X(r), P_Y(r)) = 1$ in $\mathbb{Z}_p$ with probability 1 over the choice of prime p (since the polynomials are coprime, their resultant is nonzero).

If $X \cap Y \neq \emptyset$, let $k \in X \cap Y$. Then both $P_X(k) = 0$ and $P_Y(k) = 0$. The probability that $r = k \bmod p$ is at most $1/p = O(1/n^2)$ for any fixed k. By a union bound over at most n possible values of $k \in X \cap Y$, the probability that r equals some element of $X \cap Y$ modulo p is at most $n/p = O(1/n)$.

When $r \notin X \cap Y \pmod p$, at least one of $P_X(r)$ or $P_Y(r)$ is nonzero modulo p, but this does not immediately give us the desired bound. Instead, we use a more refined analysis:

If $X \cap Y \neq \emptyset$, then $\gcd(P_X, P_Y)$ has degree at least $|X \cap Y| \geq 1$. The key observation is that $P_X(r) \equiv 0 \pmod{p}$ if and only if r is a root of P_X modulo p. Since P_X has degree at most n, it has at most n roots modulo p. Therefore, $\Pr[P_X(r) = 0] \leq n/p = O(1/n)$.

By repeating this protocol $O(\log n)$ times independently with independent public randomness, we can reduce the error probability to $O(1/n^c)$ for any constant c. Each iteration requires Alice to send $O(\log p) = O(\log n)$ bits.

Therefore, the total communication is $O(\log n \cdot \log n) = O(\log^2 n)$ bits.

For the improved bound of $O(\log n)$, we use a more sophisticated fingerprinting scheme based on modular arithmetic. Specifically, we use the fact that checking whether $X \cap Y = \emptyset$ is equivalent to checking whether $\sum_{i \in X} r^i$ and $\sum_{j \in Y} r^j$ share a common factor structure, which can be done with $O(\log n)$ bits of communication using appropriate field arithmetic and error reduction.

Since $R(\sigma) \leq R^{\text{pub}}(\sigma)$ by definition (private randomness is at least as powerful as public randomness), we conclude that $R(\sigma) = O(\log n)$. □

E.7.4 Gemini

Proof. We interpret σ as the Greater Than function (GT), where $\text{GT}(x, y) = 1$ if and only if the integer represented by x is greater than y. This problem reduces to determining the first index i where the bit strings x and y differ (the Most Significant Bit principle). If such an i exists, $x > y$ if and only if $x_i = 1$ and $y_i = 0$.

Part 1: Public Coin Protocol ($R^{\text{pub}}(\sigma) = O(\log n)$) The problem is finding the smallest index $i \in \{1, \ldots, n\}$ such that $x_i \neq y_i$. This can be modeled as searching for an element in the domain $\{1, \ldots, n\}$. We employ a binary search strategy. At any step with a candidate interval $[L, R]$, let $M = \lfloor (L + R)/2 \rfloor$. Alice and Bob need to determine if $x_{[L,M]} = y_{[L,M]}$. Using public coins, Alice and Bob can check the equality of two k-bit substrings with error probability at most $1/3$ using $O(1)$ bits of communication (e.g., by comparing random fingerprints over a field or using a shared random hash function). This reduction maps the communication problem to the problem of *binary search with noisy comparisons*. It is a known result (e.g., Feige et al., 1994) that searching for a target in a domain of size n using comparison queries that return the correct direction with probability $p > 1/2$ requires $O(\log n)$ queries to achieve a constant failure probability. Since each query (equality check) costs $O(1)$ bits, the total communication complexity is $O(\log n)$. Thus, $R^{\text{pub}}(\sigma) = O(\log n)$.

(Note: The "easier task" bound $O(\log n \log \log n)$ is achieved by a standard binary search where the error probability at step j is set to $\delta_j \approx 1/(\log n)^2$, requiring $O(\log \delta_j^{-1}) = O(\log \log n)$ bits per step. Summing over $\log n$ steps yields the looser bound.)

Part 2: Deterministic/Private Coin Complexity ($R(\sigma) = O(\log n)$) We appeal to Newman's Theorem, which relates public-coin and private-coin randomized communication complexities. For any function f with inputs of length n and error parameter ε, the private-coin randomized complexity $R_\varepsilon(f)$ satisfies:

$$R_\varepsilon(f) \leq R^{\text{pub}}_{\varepsilon'}(f) + O(\log n + \log \varepsilon^{-1}).$$

Using the result from Part 1 where $R^{\text{pub}}(\sigma) = O(\log n)$ with constant error, we obtain:

$$R(\sigma) \leq O(\log n) + O(\log n) = O(\log n).$$

This improves upon the $O(\log^2 n)$ bound typically derived from a naive noisy binary search without public randomness (Exercise 3.10). □

E.7.5 Grok

Proof. The function σ is the greater-than function on n-bit inputs: Alice has $x \in \{0,1\}^n$, Bob has $y \in \{0,1\}^n$, both interpreted as integers in $[0, 2^n - 1]$, and $\sigma(x, y) = 1$ if the integer for x is strictly greater than that for y.

To show $R^{\text{pub}}(\sigma) = O(\log n)$.

The protocol proceeds as follows. The public coins are used to select parameters for fingerprinting at doubling prefix lengths.

The prefix lengths are $m_j = 2^j$ for $j = 0, 1, \ldots, \log n$ (up to the largest $2^j \leq n$).

There are $O(\log n)$ such lengths.

For each m_j, the public coins select an independent random prime $p_j = O(m_j)$ and random $r_j \mod p_j$.

For each j, Alice computes the fingerprint of her prefix of length m_j evaluated at $r_j \mod p_j$, but to save communication, the protocol is adaptive or optimized.

The sophisticated protocol uses a tree structure to group the prefix tests and share computations, ensuring the total communication is $O(\log n)$.

The tree is a binary decision tree on the bit positions, and the tests are performed in a way that the cumulative cost of fingerprint computations and transmissions is $O(\log n)$ bits in total, with the error analysis showing constant error probability.

The details of the tree construction ensure that the information from the noisy equality tests is sufficient to identify the determining bit with high probability.

Thus, the protocol has communication $O(\log n)$ and constant error.

By Newman's theorem (which adds $O(\log n)$ communication for the input length n), there is a private-coin protocol with $O(\log n)$ communication and constant error.

This improves the bound from Exercise 3.10.

For the easier task, a protocol with $O(\log n \log \log n)$ communication is obtained by repeating each equality test $O(\log \log n)$ times in the doubling scheme or simple binary search, ensuring low error per test and total cost $O(\log n \log \log n)$.

□

E.8 Problem 8

E.8.1 Problem Statement

Theorem E.8. *Show that a similar connection holds for zero error randomized complexity. That is, for a distribution μ, denote by $\mathcal{L}_R^0(\ell)$ the expected communication used by a deterministic protocol for f (expectation taken over all inputs, weighted by μ). Prove that $R_0^{\mu,pub}(f) = \max_\ell \mathcal{L}_R^0(\ell)$.*

E.8.2 GPT

Proof. Let f be a function, and let μ be a distribution over the inputs. We are given that $\mathcal{L}_R^0(\ell)$ denotes the expected communication used by a deterministic protocol for f with error 0, where the expectation is taken over all inputs, weighted by μ. We are tasked with proving that

$$R_0^{\mu,\text{pub}}(f) = \max_\ell \mathcal{L}_R^0(\ell).$$

We begin by recalling the definition of $R_0^{\mu,\text{pub}}(f)$. It represents the zero-error randomized communication complexity of the function f with respect to the distribution μ, where the expectation is taken over the randomness of the protocol. Specifically,

$$R_0^{\mu,\text{pub}}(f) = \min_P \max_{\ell \in \mathcal{L}} \mathbb{E}_{x \sim \mu}[C_P(x, \ell)],$$

where P is a randomized protocol, and $C_P(x, \ell)$ is the communication cost for the input x and label ℓ under protocol P.

Now, consider the deterministic protocol for f that minimizes the expected communication, denoted by $\mathcal{L}_R^0(\ell)$. By definition, we have

$$\mathcal{L}_R^0(\ell) = \mathbb{E}_{x \sim \mu}[C_D(x, \ell)],$$

where $C_D(x, \ell)$ is the communication cost of the deterministic protocol for the input x and label ℓ. Since the zero-error randomized communication complexity $R_0^{\mu,\text{pub}}(f)$ is the expected communication of a randomized protocol, it is at most the communication complexity of the best deterministic protocol for each ℓ. Therefore, for all ℓ, we have

$$R_0^{\mu,\text{pub}}(f) \leq \mathcal{L}_R^0(\ell).$$

Next, we observe that the zero-error randomized communication complexity can be achieved by a randomized protocol that uses at most the maximum communication cost over all ℓ. That is, there exists a randomized protocol P such that

$$R_0^{\mu,\text{pub}}(f) = \max_\ell \mathbb{E}_{x\sim\mu}[C_P(x,\ell)].$$

Since the communication cost of P for each ℓ is at most $\mathcal{L}_R^0(\ell)$, we obtain

$$R_0^{\mu,\text{pub}}(f) \leq \max_\ell \mathcal{L}_R^0(\ell).$$

Conversely, for any deterministic protocol D, we have that

$$\mathcal{L}_R^0(\ell) = \mathbb{E}_{x\sim\mu}[C_D(x,\ell)],$$

and by definition of the zero-error randomized communication complexity, the randomized protocol cannot use less communication than the best deterministic protocol for any input distribution. Therefore, we also have

$$\max_\ell \mathcal{L}_R^0(\ell) \leq R_0^{\mu,\text{pub}}(f).$$

Combining both inequalities, we conclude that

$$R_0^{\mu,\text{pub}}(f) = \max_\ell \mathcal{L}_R^0(\ell).$$

This completes the proof. □

E.8.3 Claude

Proof. We prove that $R_0^{\mu,\text{pub}}(f) = \max_\ell \mathcal{L}_R^0(\ell)$, where $R_0^{\mu,\text{pub}}(f)$ denotes the zero-error randomized communication complexity of f with respect to distribution μ and public randomness, and $\mathcal{L}_R^0(\ell)$ denotes the expected communication of a deterministic protocol ℓ under distribution μ.

First, we establish that $R_0^{\mu,\text{pub}}(f) \geq \max_\ell \mathcal{L}_R^0(\ell)$.

Let Π be an optimal zero-error randomized protocol with public randomness achieving $R_0^{\mu,\text{pub}}(f)$. By definition, Π uses a distribution over deterministic protocols, say $\{\ell_i\}_{i\in I}$ with probabilities $\{p_i\}_{i\in I}$, where each ℓ_i computes f correctly on all inputs. The expected communication cost of Π under distribution μ is

$$R_0^{\mu,\text{pub}}(f) = \sum_{i\in I} p_i \cdot \mathcal{L}_R^0(\ell_i).$$

For any particular deterministic protocol ℓ_j in the support of Π, we have $\mathcal{L}_R^0(\ell_j) \leq R_0^{\mu,\text{pub}}(f)/p_j$. However, since Π is optimal and uses public randomness, we have

$$R_0^{\mu,\text{pub}}(f) = \sum_{i\in I} p_i \cdot \mathcal{L}_R^0(\ell_i) \geq \min_{i\in I} \mathcal{L}_R^0(\ell_i).$$

But more importantly, for any deterministic zero-error protocol ℓ for f, we have $\mathcal{L}_R^0(\ell) \geq R_0^{\mu,\text{pub}}(f)$ would be incorrect. Instead, we note that $R_0^{\mu,\text{pub}}(f) \leq \mathcal{L}_R^0(\ell)$ for any deterministic protocol ℓ.

Let us reconsider the argument. For the lower bound, observe that any zero-error randomized protocol with public randomness is a probability distribution over deterministic zero-error protocols. If Π achieves $R_0^{\mu,\text{pub}}(f)$, then

$$R_0^{\mu,\text{pub}}(f) = \mathbb{E}_{\text{rand}}[\mathbb{E}_\mu[\text{cost}(\Pi,x)]] = \sum_i p_i \mathcal{L}_R^0(\ell_i) \geq \min_i \mathcal{L}_R^0(\ell_i).$$

For the upper bound, $R_0^{\mu,\text{pub}}(f) \leq \max_\ell \mathcal{L}_R^0(\ell)$, consider the randomized protocol that selects the optimal deterministic protocol ℓ^* achieving $\max_\ell \mathcal{L}_R^0(\ell)$ with probability 1. This gives $R_0^{\mu,\text{pub}}(f) \leq \mathcal{L}_R^0(\ell^*)$.

For the reverse inequality, note that any randomized protocol is a convex combination of deterministic protocols. By Yao's minimax principle for zero-error protocols, we have

$$R_0^{\mu,\text{pub}}(f) = \min_{\text{rand. protocol}} \mathbb{E}_\mu[\text{cost}] = \min_{\ell \text{ det.}} \mathcal{L}_R^0(\ell).$$

Actually, we need to reconsider: for zero-error protocols, the correct statement is that the best randomized protocol performs at least as well as the average over deterministic protocols, but we seek the worst-case deterministic protocol. By Yao's principle for distributional complexity,

$$R_0^{\mu,\text{pub}}(f) = \min_{\ell \text{ deterministic}} \mathcal{L}_R^0(\ell) = \max_\ell \mathcal{L}_R^0(\ell)$$

where the maximum is taken over all deterministic zero-error protocols, which by Yao's minimax theorem equals the minimum expected cost over all randomized protocols. □

E.8.4 Gemini

Proof. We aim to establish the relationship between the public-coin zero-error randomized communication complexity, denoted as $R_0^{\text{pub}}(f)$, and the distributional zero-error complexity. Let $\mathcal{P}$ be the set of all deterministic protocols that compute the function f with zero error. Since the domain of inputs $X \times Y$ is finite, we may assume without loss of generality that $\mathcal{P}$ is finite (considering only optimal protocols).

The public-coin zero-error randomized complexity $R_0^{\text{pub}}(f)$ is defined as the minimum expected cost of a protocol, where the minimum is taken over all probability distributions λ over $\mathcal{P}$, and the cost is maximized over all inputs (x, y). Formally:

$$R_0^{\text{pub}}(f) = \min_\lambda \max_{(x,y)\in X\times Y} \sum_{P\in\mathcal{P}} \lambda(P) \cdot \text{cost}(P, x, y).$$

For a fixed distribution μ over the inputs $X \times Y$, the distributional zero-error complexity, denoted here as $D_0^\mu(f)$ (or $\mathcal{L}_R^0(\mu)$ in the problem statement context), is the minimum expected cost of a deterministic zero-error protocol with respect to μ. Formally:

$$D_0^\mu(f) = \min_{P\in\mathcal{P}} \sum_{(x,y)\in X\times Y} \mu(x, y) \cdot \text{cost}(P, x, y).$$

We model this scenario as a zero-sum game between a Protocol Designer and an Adversary.

- The Designer chooses a deterministic zero-error protocol $P \in \mathcal{P}$.
- The Adversary chooses an input $(x, y) \in X \times Y$.
- The payoff (cost) is given by the communication cost function $M(P, (x, y)) = \text{cost}(P, x, y)$.

The Designer aims to minimize the cost, while the Adversary aims to maximize it. We extend this to mixed strategies: the Designer chooses a distribution λ over $\mathcal{P}$, and the Adversary chooses a distribution μ over $X \times Y$.

According to von Neumann's Minimax Theorem, for any finite two-player zero-sum game, the minimax value equals the maximin value:

$$\min_\lambda \max_\mu \mathop{\mathbb{E}}_{\substack{P\sim\lambda \\ (x,y)\sim\mu}} [M(P, (x, y))] = \max_\mu \min_\lambda \mathop{\mathbb{E}}_{\substack{P\sim\lambda \\ (x,y)\sim\mu}} [M(P, (x, y))].$$

The left-hand side simplifies to the randomized complexity:

$$\min_\lambda \max_{(x,y)} \mathbb{E}_{P\sim\lambda}[\text{cost}(P, x, y)] = R_0^{\text{pub}}(f),$$

since for a fixed λ, the Adversary maximizes the expectation by choosing a pure strategy (a specific input (x, y)) that maximizes the cost.

The right-hand side simplifies to the maximum distributional complexity:

$$\max_{\mu} \min_{P} \mathbb{E}_{(x,y)\sim\mu}[\text{cost}(P, x, y)] = \max_{\mu} D_0^{\mu}(f),$$

since for a fixed μ, the Designer minimizes the expectation by choosing a pure strategy (a specific protocol P) that minimizes the average cost.

Thus, we obtain the equality:

$$R_0^{\text{pub}}(f) = \max_{\mu} D_0^{\mu}(f).$$

This proves that the zero-error randomized communication complexity is exactly equal to the maximum distributional zero-error complexity over all input distributions. □

E.8.5 Grok

Proof. A public-coin zero-error randomized protocol for f corresponds to a distribution over random strings r, where for each r the fixed-r behavior defines a deterministic protocol ℓ_r that computes f correctly on every input.

The expected communication cost under μ is $\mathbb{E}_r[\mathcal{L}_R^0(\ell_r)] = \mathbb{E}_r\left[\mathbb{E}_{(x,y)\sim\mu}[\text{cc}(\ell_r, x, y)]\right]$.

Since this is a convex combination of the values $\mathcal{L}_R^0(\ell_r)$ over correct deterministic protocols ℓ_r, it holds that $\mathbb{E}_r[\mathcal{L}_R^0(\ell_r)] \geq \min_\ell \mathcal{L}_R^0(\ell)$, where the minimum is over all correct deterministic protocols ℓ for f.

Thus, $R_0^{\mu,\text{pub}}(f) \geq \min_\ell \mathcal{L}_R^0(\ell)$.

For the reverse direction, consider the correct deterministic protocol ℓ^* that achieves the minimum $\mathcal{L}_R^0(\ell^*)$. A public-coin protocol can simply ignore the public coins and always run ℓ^*, yielding expected communication cost $\mathcal{L}_R^0(\ell^*)$ under μ.

Thus, $R_0^{\mu,\text{pub}}(f) \leq \min_\ell \mathcal{L}_R^0(\ell)$.

The equality follows.

□

E.9 Problem 9

E.9.1 Problem Statement

Problem E.9 (Open Problem 3.23)**.** *How far can the best lower bound obtained using this technique be from the true randomized complexity?*

E.9.2 GPT

Proof. Let the true randomized complexity of the problem be denoted by C_{true}. Let the best lower bound obtained using the technique be denoted by C_{lb}.

We seek to determine how far the best lower bound C_{lb} can be from the true randomized complexity C_{true}.

By definition, the lower bound C_{lb} satisfies the condition that for any randomized algorithm A solving the problem, the expected running time of A is at least C_{lb}. That is, for any A, we have:

$$T(A) \geq C_{lb}$$

where $T(A)$ represents the expected running time of algorithm A.

The true randomized complexity C_{true} is the infimum of the expected running times of all randomized algorithms solving the problem. Formally, we have:

$$C_{true} = \inf_{A} T(A)$$

It follows that $C_{lb} \leq C_{true}$ because C_{lb} is a lower bound for the expected running time of any algorithm, and C_{true} is the smallest such lower bound.

Next, consider the possibility that the best lower bound C_{lb} is strictly smaller than C_{true}. This would imply that there exists a randomized algorithm A with expected running time $T(A) > C_{lb}$ but arbitrarily close to C_{true}. In this case, the distance between C_{lb} and C_{true} would be the difference:

$$C_{true} - C_{lb}$$

Since C_{lb} is a lower bound and C_{true} is the infimum of the expected running times, the best lower bound C_{lb} can never exceed the true randomized complexity C_{true}, meaning that:

$$C_{lb} \leq C_{true}$$

Thus, the largest possible distance between C_{lb} and C_{true} is exactly $C_{true} - C_{lb}$.

Finally, to conclude, the best lower bound obtained using this technique can be at most C_{true} away from the true randomized complexity. Therefore, the best lower bound C_{lb} can be at most C_{true} units away from the true randomized complexity.

□

E.9.3 Claude

Proof. This is an open problem regarding the gap between lower bounds obtained via a specific technique (typically the distributional complexity method or related approaches) and the true randomized complexity. We provide a rigorous analysis of what is currently known about this gap.

Let $R(f)$ denote the randomized query complexity of a Boolean function f, and let $LB(f)$ denote the best lower bound obtainable using the technique in question (typically the corruption bound or distributional complexity method).

We establish the following results:

First, we observe that by definition, for any Boolean function f, we have $LB(f) \leq R(f)$, since any valid lower bound technique cannot exceed the true complexity.

Second, we demonstrate that there exist functions where the gap is at most logarithmic. Specifically, for the distributional complexity method applied to function $f : \{0,1\}^n \to \{0,1\}$, if $D^\mu(f)$ denotes the distributional complexity under distribution μ, then by Yao's minimax principle, we have

$$R(f) = \min_{\mathcal{R}} \max_{\mu} D^\mu(f) = \max_{\mu} \min_{\mathcal{R}} D^\mu(f)$$

where $\mathcal{R}$ ranges over randomized algorithms and μ ranges over input distributions.

Third, we note that for certain specific techniques such as the corruption bound method, there exist explicit examples showing separations. Consider a function f where the corruption bound yields $LB(f) = \Omega(\sqrt{n})$ while $R(f) = O(\sqrt{n} \log n)$, demonstrating a logarithmic gap.

Fourth, we observe that no super-logarithmic separation is currently known for the standard corruption bound technique when applied to total Boolean functions. That is, there is no known example where $R(f)/LB(f) = \omega(\log n)$.

Finally, we conclude that the question of whether there exists a constant $c > 0$ such that $R(f) \leq c \cdot LB(f)$ for all Boolean functions f remains open. Similarly, whether there exists a function f with $R(f)/LB(f) = \omega(\log n)$ is unknown.

Therefore, the precise characterization of the maximum possible gap between the lower bound technique and true randomized complexity remains an open problem, with current knowledge suggesting the gap is at most polylogarithmic but no tight bounds established. □

E.9.4 Gemini

Proof. We prove the Discrepancy Lower Bound, which is the technique referenced in the problem. Let $f : X \times Y \to \{0,1\}$ be a Boolean function, and let μ be a probability distribution over the input space $X \times Y$. The discrepancy of f with respect to μ is defined as:

$$\text{disc}_\mu(f) = \max_R \left| \sum_{(x,y) \in R} \mu(x,y)(-1)^{f(x,y)} \right|,$$

where the maximum is taken over all combinatorial rectangles $R \subseteq X \times Y$.

Let Π be a deterministic protocol that computes f with distributional error ε under μ. Let $C(\Pi)$ denote the communication cost of Π. The protocol partitions the input space into a set of disjoint combinatorial rectangles $\mathcal{R} = \{R_1, \ldots, R_{2^{C(\Pi)}}\}$. For each rectangle R_i, the protocol outputs a constant value $v_i \in \{0, 1\}$.

The condition that Π has error at most ε means:

$$\Pr_{\mu}[\Pi(x, y) \neq f(x, y)] \leq \varepsilon \implies \Pr_{\mu}[\Pi(x, y) = f(x, y)] \geq 1 - \varepsilon.$$

Consider the advantage of the protocol, defined as the correlation between the protocol output and the function value:

$$\sum_{(x,y) \in X \times Y} \mu(x, y)(-1)^{f(x,y) \oplus \Pi(x,y)}.$$

Since $(-1)^{a \oplus b} = 1$ if $a = b$ and -1 if $a \neq b$, we can rewrite this sum as:

$$\Pr[\Pi = f] \cdot (1) + \Pr[\Pi \neq f] \cdot (-1) \geq (1 - \varepsilon) - \varepsilon = 1 - 2\varepsilon.$$

We can also compute this sum by iterating over the rectangles R_i defined by the protocol. Since $\Pi(x, y) = v_i$ for all $(x, y) \in R_i$:

$$\sum_{(x,y) \in X \times Y} \mu(x, y)(-1)^{f(x,y) \oplus \Pi(x,y)} = \sum_{i=1}^{2^{C(\Pi)}} \sum_{(x,y) \in R_i} \mu(x, y)(-1)^{f(x,y) \oplus v_i}.$$

Using the property $(-1)^{A \oplus B} = (-1)^A (-1)^B$, we have:

$$\sum_{i=1}^{2^{C(\Pi)}} (-1)^{v_i} \sum_{(x,y) \in R_i} \mu(x, y)(-1)^{f(x,y)}.$$

By the definition of discrepancy, for any rectangle R_i, we have:

$$\left| \sum_{(x,y) \in R_i} \mu(x, y)(-1)^{f(x,y)} \right| \leq \operatorname{disc}_{\mu}(f).$$

Therefore, the magnitude of the total sum is bounded by:

$$\left| \sum_{i=1}^{2^{C(\Pi)}} (-1)^{v_i} \sum_{(x,y) \in R_i} \mu(x, y)(-1)^{f(x,y)} \right| \leq \sum_{i=1}^{2^{C(\Pi)}} \operatorname{disc}_{\mu}(f) = 2^{C(\Pi)} \operatorname{disc}_{\mu}(f).$$

Combining the lower bound from the accuracy requirement and the upper bound from the discrepancy:

$$1 - 2\varepsilon \leq 2^{C(\Pi)} \operatorname{disc}_{\mu}(f).$$

Solving for the communication cost $C(\Pi)$:

$$2^{C(\Pi)} \geq \frac{1 - 2\varepsilon}{\operatorname{disc}_{\mu}(f)} \implies C(\Pi) \geq \log_2 \left(\frac{1 - 2\varepsilon}{\operatorname{disc}_{\mu}(f)} \right).$$

By Yao's Minimax Principle, the randomized public-coin complexity $R_{\varepsilon}^{pub}(f)$ is equal to the maximum distributional complexity $D_{\varepsilon}^{\mu}(f)$ over all distributions μ. Thus:

$$R_{\varepsilon}^{pub}(f) \geq \log_2 \left(\frac{1 - 2\varepsilon}{\operatorname{disc}_{\mu}(f)} \right) = \Omega \left(\log \frac{1}{\operatorname{disc}_{\mu}(f)} \right).$$

This establishes the lower bound technique referenced in the problem. The open problem asks whether this lower bound is tight (up to polynomial factors); it is now known that the gap is at most polynomial, specifically related to the logarithm of the partition bound or the γ_2 norm. $\square$

E.9.5 Grok

Proof. The technique referred to is the corruption method for proving lower bounds on the bounded-error randomized communication complexity $R(f)$.

The corruption bound provides a lower bound $\mathrm{corr}_\epsilon(f)$ such that $R_\epsilon(f) \geq \mathrm{corr}_\epsilon(f)$.

The corruption method is powerful and yields asymptotically tight $\Omega(n)$ lower bounds for the disjointness function, where $R(\mathrm{DISJ}_n) = \Theta(n)$.

However, the question of the largest possible gap between the true randomized complexity $R(f)$ and the best lower bound obtainable via the corruption method remains unresolved.

Specifically, it is open whether there always exists a polynomial p such that $R(f) \leq p(\mathrm{corr}(f))$ for all functions f (up to constant factors in the error parameter ϵ and additive constants), or whether superpolynomial gaps exist.

No larger-than-polynomial gaps are known, and the problem of determining the maximal gap is open. □

E.10 Problem 10

E.10.1 Problem Statement

Theorem E.10 (Exercise 3.25)**.** *Prove that $R^{II}(DISJ) = O(\sqrt{n \log n})$. Contrast this result with the fact that $R(DISJ) = \Theta(n)$ (Example 3.22). For the uniform distribution (which is rectangular), show that $D^{uniform}(DISJ) = \Theta(\sqrt{n})$.*

E.10.2 GPT

Proof. We are given that $R^{II}(\mathrm{DISJ}) = O(\sqrt{n \log n})$ and asked to contrast this with the fact that $R(\mathrm{DISJ}) = \Theta(n)$. Additionally, we are asked to show that for the uniform distribution (which is rectangular), $D^{\text{uniform}}(\mathrm{DISJ}) = \Theta(\sqrt{n})$.

First, we will begin with the analysis of $R^{II}(\mathrm{DISJ})$ and show that it is indeed $O(\sqrt{n \log n})$.

Step 1: Upper Bound for $R^{II}(\mathrm{DISJ})$ Recall that $R^{II}(\mathrm{DISJ})$ is the randomized decision tree complexity for the DISJ function under the II model. In the II model, the decision tree is allowed to make random queries, and the complexity of a decision tree is determined by the expected depth of the tree.

The disjointness problem DISJ asks whether the intersection of two sets, each of size n, is empty. It is known that in the worst-case, a decision tree for this problem requires $\Theta(n)$ queries in the deterministic case.

However, when we allow randomization (i.e., in the R^{II} model), we can significantly reduce the number of queries. In fact, the randomized complexity $R^{II}(\mathrm{DISJ})$ has been shown to be upper-bounded by $O(\sqrt{n \log n})$. This result follows from the fact that randomization allows the decision tree to probe the input more efficiently, utilizing the probabilistic structure of the inputs to reduce the depth of the decision tree. Specifically, the use of randomization allows the tree to eliminate large portions of the input space with each query, leading to an expected depth that scales more favorably compared to the deterministic case.

Thus, we have:

$$R^{II}(\mathrm{DISJ}) = O(\sqrt{n \log n}).$$

Step 2: Comparison with $R(\mathrm{DISJ}) = \Theta(n)$ Next, we compare this result with the deterministic complexity $R(\mathrm{DISJ})$. In the deterministic case, the decision tree must examine every input element in the worst case, leading to a complexity of $\Theta(n)$ for the disjointness problem. This is significantly worse than the randomized case, where randomization reduces the number of queries needed to $O(\sqrt{n \log n})$.

Thus, the result $R^{II}(\mathrm{DISJ}) = O(\sqrt{n \log n})$ demonstrates the power of randomization in reducing the query complexity compared to the deterministic case.

Step 3: Lower Bound for $D^{\text{uniform}}(\mathrm{DISJ})$ Finally, we consider the uniform distribution case, where the inputs are drawn from a uniform distribution. In this case, the complexity of the decision tree under the uniform distribution is denoted by $D^{\text{uniform}}(\mathrm{DISJ})$.

For the disjointness problem, it has been shown that under the uniform distribution, the expected depth of the decision tree is $\Theta(\sqrt{n})$. This result follows from the fact that the uniform distribution provides a balanced structure to the input, which allows the decision tree to efficiently partition the input space using random queries. Specifically, since each query provides a good amount of information about the input, the number of queries required is reduced compared to the worst-case deterministic complexity.

Thus, we have:

$$D^{\text{uniform}}(\text{DISJ}) = \Theta(\sqrt{n}).$$

Conclusion We have shown that $R^{II}(\text{DISJ}) = O(\sqrt{n \log n})$, which is a significant improvement over the deterministic complexity of $\Theta(n)$. We also showed that for the uniform distribution, the complexity of the decision tree is $D^{\text{uniform}}(\text{DISJ}) = \Theta(\sqrt{n})$. This highlights the efficiency gains achievable through randomization and the structure provided by the uniform distribution. □

E.10.3 Claude

Proof. We prove the three parts of the theorem in order.

Part 1: $R^{II}(\textbf{DISJ}) = O(\sqrt{n \log n})$.

Let Alice and Bob each hold n-bit strings $x, y \in \{0,1\}^n$ respectively, and they wish to determine if $\text{DISJ}(x, y) = 1$, i.e., whether $x \wedge y = 0$.

We use a protocol based on sampling. Divide the protocol into $T = O(\sqrt{n \log n})$ rounds. In each round, Alice samples an index i uniformly at random from $\{1, \ldots, n\}$ and sends i to Bob (using $\lceil \log n \rceil$ bits). Bob checks if $x_i \wedge y_i = 1$. If so, Bob announces that the sets intersect and the protocol terminates. Otherwise, they continue to the next round.

After T rounds, if no intersection has been found, they declare that the sets are disjoint.

For the analysis, suppose $x \wedge y \neq 0$, so there exists at least one index j where $x_j = y_j = 1$. The probability that a single random sample misses all such indices is at most $1 - \frac{1}{n}$. After T independent samples, the probability of missing all intersections is at most $(1 - \frac{1}{n})^T$.

Setting $T = c\sqrt{n \log n}$ for sufficiently large constant c, we have:

$$(1 - \frac{1}{n})^T \leq e^{-T/n} = e^{-c\sqrt{\log n/n}} \cdot e^{-c\sqrt{\log n/n} \cdot (\sqrt{n}-1)} \leq e^{-c\sqrt{n \log n}/n} = e^{-c\sqrt{\log n/n}}.$$

For c sufficiently large and n sufficiently large, this error probability can be made arbitrarily small (say, at most $1/3$).

The total communication is $T \cdot \lceil \log n \rceil = O(\sqrt{n \log n} \cdot \log n) = O(\sqrt{n \log n} \cdot \log n)$.

However, using a more sophisticated protocol based on the birthday paradox and hashing, we can achieve $O(\sqrt{n \log n})$ communication. Alice sends a random sample of $O(\sqrt{n})$ indices from her support, and Bob checks for collisions. With appropriate error correction using $O(\log n)$ repetitions, the total communication is $O(\sqrt{n \log n})$.

Part 2: Contrast with $R(\textbf{DISJ}) = \Theta(n)$.

In the standard randomized model with two-sided error, it is known (Example 3.22) that $R(\text{DISJ}) = \Theta(n)$. This shows that the rectangular distribution assumption (product distribution) in the R^{II} model provides significant advantage, reducing communication from linear to sublinear. The key difference is that in R^{II}, the distribution over inputs is a product distribution, which allows for more efficient sampling-based protocols.

Part 3: $D^{\textbf{uniform}}(\textbf{DISJ}) = \Theta(\sqrt{n})$.

For the uniform distribution over all pairs $(x, y) \in \{0,1\}^n \times \{0,1\}^n$, we establish both upper and lower bounds.

Upper bound: Alice sends a random subset S of her 1-indices of size $|S| = O(\sqrt{n})$ to Bob. Bob checks if any index in S is also a 1-index in y. If so, they intersect. If not, they declare disjoint with small error probability. The expected number of 1-indices in x is $n/2$, and similarly for y. The expected size of the intersection is $n/4$. By sampling $O(\sqrt{n})$ indices from Alice's 1-positions, the probability of missing all intersection points (when the intersection is non-empty) can be bounded using concentration inequalities. The communication is $O(\sqrt{n} \log n)$ bits, which gives $D^{\text{uniform}}(\text{DISJ}) = O(\sqrt{n})$ when measured in bits.

Lower bound: Under the uniform distribution, the expected intersection size is $\Theta(n)$. Using information-theoretic arguments and the fact that Alice must communicate enough information to allow Bob to distinguish between disjoint and intersecting sets with constant advantage, we require $\Omega(\sqrt{n})$ communication. This follows from discrepancy bounds and the fact that the uniform distribution has specific structure that prevents protocols with $o(\sqrt{n})$ communication from achieving constant success probability.

Therefore, $D^{\text{uniform}}(\text{DISJ}) = \Theta(\sqrt{n})$. □

E.10.4 Gemini

Proof. We analyze the communication complexity of the disjointness function DISJ_n under product distributions.

Part 1: Upper bound for Product Distributions Let μ be an arbitrary product distribution on $\{0,1\}^n \times \{0,1\}^n$. We show that there exists a protocol with communication complexity $O(\sqrt{n \log n})$ that achieves low error under μ. Let $k = \sqrt{n \ln(2/\varepsilon)}$ for a fixed error parameter ε. Consider the following simple protocol Π: Alice checks if the Hamming weight $|x| \leq k$.

1. If $|x| \leq k$, Alice sends the indices of the 1s in x to Bob. This requires $O(k \log n) = O(\sqrt{n \log n})$ bits. Bob then checks if $x \cap y = \emptyset$ and outputs the result.

2. If $|x| > k$, Alice sends a special "heavy" symbol.

3. If Bob receives the "heavy" symbol, he checks if $|y| \leq k$.

4. If $|y| \leq k$, Bob sends the indices of y to Alice (cost $O(\sqrt{n \log n})$). Alice checks intersection.

5. If both $|x| > k$ and $|y| > k$, the protocol outputs 1 (indicating intersection).

We analyze the error of this protocol under μ. The protocol is correct whenever $|x| \leq k$ or $|y| \leq k$. The only potential error occurs when $|x| > k$, $|y| > k$, and $x \cap y = \emptyset$. Since μ is a product distribution, the bits x_i are independent, and y_i are independent. Babai, Frankl, and Simon (1986) showed that for any product distribution, if the probability of the sets being "large" is significant, then the probability of them being disjoint is small. Specifically, if $\Pr_\mu[|x| > k] > \varepsilon$ and $\Pr_\mu[|y| > k] > \varepsilon$, and we assume the distribution is such that disjointness is possible, it can be shown that the error in assuming intersection for large sets is bounded. However, a more direct argument for the specific "hard" distributions (where $p_i \approx 1/\sqrt{n}$) suffices to convey the intuition. If probabilities were higher, intersection would be certain. If probabilities are lower, sets are small. The threshold $k \approx \sqrt{n}$ balances these cases. The bound $O(\sqrt{n \log n})$ holds for the randomized complexity with two-sided error over any product distribution ($R^{II}(\text{DISJ})$).

Part 2: Contrast with General Randomized Complexity It is a fundamental result (Example 3.22) that $R(\text{DISJ}) = \Theta(n)$. This worst-case complexity is achieved on a non-product distribution (specifically, the distribution supported on pairs (x, y) such that $|x| = |y| \approx n/2$ and $|x \cap y| \in \{0, 1\}$). The result $R^{II}(\text{DISJ}) = O(\sqrt{n \log n})$ demonstrates that disjointness is significantly easier on average (under product distributions) than in the worst case.

Part 3: The Uniform Distribution Case In the context of this problem, the "uniform distribution" refers to the symmetric product distribution ν where $\Pr[x_i = 1] = \Pr[y_i = 1] = \frac{1}{\sqrt{n}}$. Under this distribution, the expected set size is $\sqrt{n}$.

- **Upper Bound $D^\nu(\text{DISJ}) = O(\sqrt{n})$:** We construct a distributional protocol. Since ν is a product distribution, we can employ a public-coin randomized protocol and fix the random coins to obtain a deterministic one. Alice constructs a Bloom filter for her set x. Since $\mathbb{E}[|x|] = \sqrt{n}$, with high probability $|x| \leq 2\sqrt{n}$. A Bloom filter representing a set of size m with constant false positive rate requires $O(m)$ bits. Alice sends a Bloom filter of size $c\sqrt{n}$ bits. Bob queries the filter with the indices in his set y. If any query returns positive, Bob outputs 1 (intersect); otherwise 0. The one-sided error (false intersection) can be made arbitrarily small by the constant factor c. The communication cost is $O(\sqrt{n})$.

- **Lower Bound** $D^{\nu}(\text{DISJ}) = \Omega(\sqrt{n})$**:** Under ν, the probability that x and y are disjoint is approximately $(1-1/n)^n \approx e^{-1}$, which is constant. The sets x and y have size $\Theta(\sqrt{n})$ with high probability. Distinguishing disjointness for sets of size $\sqrt{n}$ is known to require $\Omega(\sqrt{n})$ bits even under this distribution. Intuitively, any protocol with cost $o(\sqrt{n})$ would fail to check a sufficient number of potential intersection points, as the "collision" space has size n and the inputs cover $\approx \sqrt{n}$ indices each. Formal lower bounds via discrepancy or volume arguments confirm that $D^{\nu}(\text{DISJ}) = \Theta(\sqrt{n})$.

□

E.10.5 Grok

Proof. We prove the three parts separately.

First, we prove that $R^{II}(\text{DISJ}) = O(\sqrt{n \log n})$.

We present a zero-error public-coin randomized protocol for DISJ with expected communication cost $O(\sqrt{n \log n})$ in the worst case.

Assume the inputs are characteristic vectors of subsets $S, T \subseteq [n]$ of size at most $k = \sqrt{n}$. (The hard case for DISJ is when the sets have size $\sqrt{n}$; for larger sets, the protocol can be adjusted accordingly, but the bound holds.)

The protocol is recursive on the current sets S, T with |S| = |T| = k'.

If $k' \leq 1$, Alice sends her set ($O(\log n)$ bits), Bob checks for intersection and outputs the result.

For larger k', using public randomness, choose a random hash $h : [n] \to [4k']$.

Alice computes the image $H_S = \{h(i) \mid i \in S\}, |H_S| \leq k'$.

She sends the sorted list of distinct elements in H_S, each requiring $O(\log k')$ bits, total $O(k' \log k')$ bits.

Bob does the same for H_T.

If $H_S \cap H_T = \emptyset$, output DISJ = 1 (disjoint).

Otherwise, let $C = H_S \cap H_T$.

Set $S' = \{i \in S \mid h(i) \in C\}, T' = \{i \in T \mid h(i) \in C\}$.

Recurse on S', T'.

The protocol is correct because if S, T disjoint, then if images intersect, S', T' remain disjoint; eventually correct at base. If not disjoint, the intersection element's hash is in C, so preserved; eventually found at base.

For the cost, the expected $|S'| = k' \cdot P(h(i) \in H_T) \leq k' \cdot (k'/4k') = k'/4$.

By linearity, expected size reduces by factor 4.

The expected number of iterations is $O(\log n)$.

The expected cost is$\sum O((k/4^j) \log(k/4^j)) = O(k \log k) = O(\sqrt{n} \log n)$.

Thus, $R^{II}(\text{DISJ}) = O(\sqrt{n \log n})$.

This contrasts with the bounded-error randomized complexity $R(\text{DISJ}) = \Theta(n)$, where error is allowed, yet the complexity is higher due to the need to handle worst-case inputs without distribution assumptions.

For the uniform distribution (rectangular), we show $D^{\text{uniform}}(\text{DISJ}) = \Theta(\sqrt{n})$.

For the upper bound, consider a deterministic protocol with communication $O(\sqrt{n})$.

Fix a set I $\subseteq [n]$ of size m = $O(\sqrt{n})$.

Alice sends her bits x_i for $i \in I$(m bits).

Bob checks if there is $i \in I$ with $x_i = y_i = 1$; if yes, output 0; else output 1.

Under uniform, $P(error) = P$(not disjoint and no overlap in I).

Since P(not disjoint) $\approx$ 1, but the conditional $P(nooverlapinI|notdisjoint) \leq P(nooverlapinI)/P(disjoint)$.

But direct: $P(notdisjointandnooverlapinI) = P(atleastoneoverlapin[n] \setminus I, nooverlapinI) \leq P(atleastoneoverlapin[n] \setminus I) \leq (n-m)/4$, no.

Each position has P(overlap) = 1/4, independent.

$P(nooverlapinI) = (3/4)^m$.

$P(notdisjoint) = 1 - (3/4)^n$.

The error is $P(notdisjoint) - P(overlapinI) = P(overlaponlyin[n] \setminus I)$.

$P(overlaponlyin[n] \setminus I) = P(nooverlapinI) - P(disjoint) = (3/4)^m - (3/4)^n \approx (3/4)^m$.

For $m = O(\sqrt{n})$, $(3/4)^{\sqrt{n}} \approx e^{-c\sqrt{n}} < 1/3$ for large n.

The protocol has error <1/3 under uniform, communication $O(\sqrt{n})$.

For the lower bound, we show $\Omega(\sqrt{n})$.

We use the rectangle argument for distributional complexity.

Under uniform μ, any deterministic protocol with error $\epsilon < 1/100$ induces a partition of the input space into monochromatic rectangles, with the 1-rectangles covering at least $(1 - 2\epsilon)$ measure of the 1-inputs.

But since $P(1) = (3/4)^n$ is tiny, but to show lower bound, we use the fixed-size case.

Following Babai, Frankl, Simon, consider the uniform μ on inputs with exactly $l = \sqrt{n}1's$ each.

This is rectangular.

We show $D^{\mu}_{\epsilon}(DISJ) = \Omega(\sqrt{n}) for \epsilon < 1/100$.

Assume a protocol with comm c, then there are 2^c rectangles.

To show lower, show that any 1-monochromatic rectangle has small size.

A 1-rectangle $F \times G$, where F, G subsets of fixed weight l vectors, and no $x \in F, y \in G$ intersect.

Suppose $|F| \geq |X|/2^{d\sqrt{n}}$, where X is all weight l vectors, $|X| = \binom{n}{l}$.

Assume d small.

Then, there is a subset $F_1 \subseteq F$, $|F_1| \geq |F|/2$, where each x in F_1 intersects at most 2ϵ fraction of y in G.

Then, find $l/3 = \sqrt{n}/3$ vectors in F_1 where each adds l/2 new elements not in the union of previous.

The number is possible because the number of x with large overlap with the current union is small.

The union has size at least $(l/3) \times (l/2) = \Omega(n)$.

Then, for G, the number is small because each y in G intersects few of the x's, so avoids large union.

The calculation shows $|G| \leq |Y|/2^{d'\sqrt{n}}$.

Thus, any large rectangle has small measure, so many rectangles needed, $c = \Omega(\sqrt{n})$.

Hence, $D^{\text{uniform}}(\text{DISJ}) = \Theta(\sqrt{n})$.

□

E.11 Problem 11

E.11.1 Problem Statement

Problem E.11. *Is* $R(f) = (R^{[1]}(f))^{O(1)}$*?*

E.11.2 GPT

Proof. We are tasked with proving whether $R(f) = (R^{[1]}(f))^{O(1)}$. To do so, we will begin by analyzing the definitions of $R(f)$ and $R^{[1]}(f)$, and investigate their relationship.

Let $R(f)$ be defined as a function that quantifies the complexity of the problem f, and let $R^{[1]}(f)$ represent a related complexity measure, possibly involving a simplified or restricted version of the problem.

The notation $R^{[1]}(f)^{O(1)}$ indicates that $R^{[1]}(f)$ is raised to a constant factor, and thus, we are considering a growth rate of $R(f)$ that is polynomially bounded by some constant multiple of $R^{[1]}(f)$.

To prove the equality, we assume that $R(f)$ and $R^{[1]}(f)$ are related by the following inequality:

$$R(f) \leq C \cdot R^{[1]}(f)$$

for some constant $C > 0$ and sufficiently large inputs. This is a standard assumption in the analysis of such problems, where the complexity of f is bounded by some constant factor of the related complexity measure $R^{[1]}(f)$.

Next, we observe that raising $R^{[1]}(f)$ to a constant factor, as indicated by the notation $R^{[1]}(f)^{O(1)}$, means that there exists a constant $k > 0$ such that:

$$R^{[1]}(f)^{O(1)} \leq C' \cdot R^{[1]}(f)$$

for some constant $C' > 0$, where k is absorbed into C'.

Thus, we have:

$$R(f) \leq C \cdot R^{[1]}(f) \leq C' \cdot R^{[1]}(f)^{O(1)}$$

This shows that $R(f)$ is bounded above by a constant multiple of $R^{[1]}(f)^{O(1)}$, which completes the proof.

Therefore, we conclude that:

$$R(f) = (R^{[1]}(f))^{O(1)}$$

as required. □

E.11.3 Claude

Proof. We prove that $R(f) = (R^{[1]}(f))^{O(1)}$ does not hold in general by constructing a counterexample.

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Recall that $R(f)$ denotes the deterministic communication complexity of f, and $R^{[1]}(f)$ denotes the one-sided error randomized communication complexity of f.

Consider the equality function $\text{EQ}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ defined by $\text{EQ}_n(x,y) = 1$ if and only if $x = y$.

For the deterministic communication complexity, it is well-known that $R(\text{EQ}_n) = n + 1$. This follows because Alice must communicate enough information to Bob so that he can determine whether $x = y$, which requires transmitting essentially all n bits of x in the worst case.

For the one-sided error randomized communication complexity, we have $R^{[1]}(\text{EQ}_n) = O(\log n)$. This can be achieved using the following protocol: Alice and Bob use a public random hash function $h : \{0,1\}^n \to \{0,1\}^{O(\log n)}$ from a pairwise independent family. Alice sends $h(x)$ to Bob, who accepts if and only if $h(x) = h(y)$. If $x = y$, then $h(x) = h(y)$ with probability 1. If $x \neq y$, then by pairwise independence, $\Pr[h(x) = h(y)] \leq 2^{-O(\log n)} < 1/3$ for an appropriate choice of range size.

Now observe that if $R(f) = (R^{[1]}(f))^{O(1)}$ held for all functions f, then for EQ_n we would have:

$$n + 1 = R(\text{EQ}_n) = (R^{[1]}(\text{EQ}_n))^{O(1)} = (O(\log n))^{O(1)} = O((\log n)^c)$$

for some constant $c > 0$ independent of n.

However, $n + 1 \neq O((\log n)^c)$ for any constant c, since n grows much faster than any polynomial in $\log n$. This yields a contradiction.

Therefore, the relation $R(f) = (R^{[1]}(f))^{O(1)}$ does not hold in general, and the answer to the question is **no**. □

E.11.4 Gemini

Proof. The answer to the problem is negative. We prove that there exists a family of functions for which the one-round (one-way) randomized communication complexity $R^{[1]}(f)$ is exponentially larger than the general randomized communication complexity $R(f)$. Specifically, we demonstrate a function f where $R(f) = O(\log n)$ and $R^{[1]}(f) = \Omega(n)$.

Consider the function $f : (\{0,1\}^n \times [n]) \times (\{0,1\}^n \times [n]) \to \{0,1\}$ defined as follows: Alice receives an input (x, i) where $x \in \{0,1\}^n$ and $i \in \{1, \ldots, n\}$. Bob receives an input (y, j) where $y \in \{0,1\}^n$ and $j \in \{1, \ldots, n\}$. The function value is given by:

$$f((x,i),(y,j)) = x_j \oplus y_i.$$

Upper bound on $R(f)$: We describe a simple two-round protocol. 1. Alice sends the index i to Bob. This requires $\lceil \log n \rceil$ bits. 2. Bob sends the bit y_i and the index j to Alice. This requires $1 + \lceil \log n \rceil$ bits. 3. Alice computes x_j, calculates $x_j \oplus y_i$, and outputs the result. The total communication is $O(\log n)$. Thus, $R(f) = O(\log n)$.

Lower bound on $R^{[1]}(f)$: The one-round complexity is defined as $R^{[1]}(f) = \min(R^{A \to B}(f), R^{B \to A}(f))$, where $R^{A \to B}$ denotes the complexity of a protocol where only Alice sends a message to Bob (and Bob outputs the result).

Case 1: Alice sends a message to Bob. Bob must compute $x_j \oplus y_i$. Since Bob holds y and i, he knows the value y_i. To determine the output, Bob must effectively compute x_j. Note that Alice holds x and

Bob holds the index j. This contains the standard INDEX problem as a subproblem (where Alice holds the database x and Bob holds the query index j). It is a standard result in communication complexity that $R^{A\to B}(\text{INDEX}) = \Omega(n)$. Even with randomized public coins, Alice must transmit $\Omega(n)$ bits of information about x to allow Bob to retrieve x_j with high probability for any j. Thus, $R^{A\to B}(f) = \Omega(n)$.

Case 2: Bob sends a message to Alice. By symmetry, Alice must compute $x_j \oplus y_i$. Alice knows x_j (once she receives j, but in a one-way protocol from Bob, she must determine the answer based on Bob's message). Equivalently, Bob holds y and Alice holds i. Bob must transmit enough information for Alice to retrieve y_i. This is again the INDEX problem with roles reversed. Thus, $R^{B\to A}(f) = \Omega(n)$.

Since both directional complexities are linear in n, we have $R^{[1]}(f) = \Omega(n)$.

Conclusion: We have shown that $R(f) = O(\log n)$ while $R^{[1]}(f) = \Omega(n)$. Since n is exponential in $\log n$, this proves that $R^{[1]}(f)$ is not bounded by any polynomial in $R(f)$. Thus, $R(f)$ and $R^{[1]}(f)$ are not polynomially equivalent. □

E.11.5 Grok

Proof. The statement is false.

Consider the family of functions $f_n = \text{INDEX}_n$, where Alice receives $x \in \{0,1\}^n$, Bob receives $i \in [n]$, and $f_n(x,i) = x_i$ (with $[n]$ encoded in binary using $\lceil \log_2(n+1) \rceil$ bits).

The randomized communication complexity $R(f_n)$ is $O(\log n)$.

A deterministic protocol achieves this: Bob sends i to Alice using $\lceil \log_2(n+1) \rceil$ bits, and Alice replies with the single bit x_i. The total communication is $O(\log n)$ with zero error.

Thus, $R(f_n) = O(\log n)$.

On the other hand, the one-way randomized communication complexity $R^{[1]}(f_n)$ (where only Alice sends a message to Bob) is $\Theta(n)$.

The upper bound is achieved by Alice sending the entire x (n bits), allowing Bob to output x_i with zero error.

The lower bound of $\Omega(n)$ holds even for randomized one-way protocols. This is a standard result in communication complexity; see for example Tim Roughgarden's notes "Communication Complexity (for Algorithm Designers)", where it is noted that the exponential separation between one-way and general communication holds for randomized protocols.

Therefore, $R(f_n) = O(\log n)$ while $R^{[1]}(f_n) = \Theta(n)$. For any fixed constant k, n^k grows faster than any multiple of $\log n$, so there is no constant k such that $O(\log n) \le C(n)^k$ for some constant C and all n. Hence, $R(f_n)$ is not bounded by $(R^{[1]}(f_n))^{O(1)}$.

This provides a counterexample showing that the statement does not hold for all f.

□

E.12 Problem 12

E.12.1 Problem Statement

Theorem E.12. *Prove that in fact,* $R_{\frac{1}{2}-\epsilon}(\text{IP}) \ge n - O(\log(1/\epsilon))$.

E.12.2 GPT

Proof. We are tasked with proving that for any $\epsilon > 0$, we have

$$R_{\frac{1}{2}-\epsilon}(\text{IP}) \ge n - O(\log(1/\epsilon)).$$

We begin by recalling that the class IP refers to the class of problems solvable by interactive proofs, and $R_{\frac{1}{2}-\epsilon}(\text{IP})$ refers to the number of rounds required in a randomized interactive proof system for a given problem from IP to achieve completeness and soundness parameters of $\frac{1}{2} - \epsilon$.

Let C denote a IP protocol for a given problem with completeness and soundness parameters $1 - \epsilon$ and ϵ, respectively. The number of rounds of communication in the protocol is denoted as r. We aim to show that for sufficiently small ϵ, the number of rounds r is at least $n - O(\log(1/\epsilon))$.

First, consider a IP protocol where the verifier performs a sequence of r challenges, each of which can be answered probabilistically by the prover. The verifier's decision is based on the number of

correct answers received, and the soundness condition ensures that a cheating prover will be caught with probability at least ϵ.

By the standard analysis of interactive proof systems, the soundness condition implies that the number of rounds r must grow with the inverse of the error probability. More specifically, we know that a randomized protocol with completeness $1 - \epsilon$ and soundness ϵ requires at least $\Omega(\log(1/\epsilon))$ rounds to achieve the desired error bounds.

Next, we consider the relationship between the number of rounds and the size of the input. The size of the input is typically proportional to n, the number of variables or the size of the problem instance. Since each round of communication requires a certain amount of resources, the number of rounds must be sufficiently large to ensure that the protocol remains secure and efficient. This implies that the number of rounds r must scale with n and $\log(1/\epsilon)$.

Thus, we conclude that for sufficiently small ϵ, the number of rounds satisfies the lower bound

$$R_{\frac{1}{2}-\epsilon}(\mathrm{IP}) \geq n - O(\log(1/\epsilon)),$$

which completes the proof. □

E.12.3 Claude

Proof. We prove that $R_{\frac{1}{2}-\epsilon}(\mathrm{IP}) \geq n - O(\log(1/\epsilon))$ where IP denotes the inner product function over $\mathbb{F}_2^n$.

Let π be any deterministic protocol computing $\mathrm{IP}(x, y) = \langle x, y \rangle = \sum_{i=1}^{n} x_i y_i \pmod 2$ with error at most $\frac{1}{2} - \epsilon$. We will show that π must communicate at least $n - O(\log(1/\epsilon))$ bits.

Consider the distributional complexity under the uniform distribution μ over $\mathbb{F}_2^n \times \mathbb{F}_2^n$. By Yao's minimax principle, it suffices to lower bound the distributional communication complexity $D_{\mu,\frac{1}{2}-\epsilon}(\mathrm{IP})$.

For any rectangle $R = A \times B \subseteq \mathbb{F}_2^n \times \mathbb{F}_2^n$, define the discrepancy:

$$\mathrm{disc}(R) = \left| \Pr_{(x,y)\sim\mu}[(x, y) \in R \mid \mathrm{IP}(x, y) = 0] - \Pr_{(x,y)\sim\mu}[(x, y) \in R \mid \mathrm{IP}(x, y) = 1] \right|$$

By the discrepancy method, any protocol with error $\frac{1}{2} - \epsilon$ must satisfy that there exists a rectangle R in the protocol partition with $\mathrm{disc}(R) \geq 2\epsilon$.

For the inner product function, we compute the discrepancy. For a rectangle $R = A \times B$, we have:

$$\mathrm{disc}(R) = \frac{1}{2^{2n}} \left| \sum_{x\in A, y\in B} (-1)^{\langle x,y \rangle} \right|$$

By Parseval's identity and Fourier analysis over $\mathbb{F}_2^n$, we can write:

$$\sum_{x\in A, y\in B} (-1)^{\langle x,y \rangle} = \sum_{S\subseteq[n]} \widehat{\mathbf{1}_A}(S)\widehat{\mathbf{1}_B}(S)$$

where $\widehat{\mathbf{1}_A}(S) = \sum_{x\in A}(-1)^{\langle x,S \rangle}$ denotes the Fourier coefficient.

By the Cauchy-Schwarz inequality:

$$\left| \sum_S \widehat{\mathbf{1}_A}(S)\widehat{\mathbf{1}_B}(S) \right| \leq \sqrt{\sum_S \widehat{\mathbf{1}_A}(S)^2} \cdot \sqrt{\sum_S \widehat{\mathbf{1}_B}(S)^2} = |A| \cdot |B|$$

Therefore, $\mathrm{disc}(R) \leq \frac{|A|\cdot|B|}{2^{2n}}$.

If the protocol uses c bits of communication, it partitions the input space into at most 2^c rectangles. For the protocol to achieve error $\frac{1}{2} - \epsilon$, at least one rectangle $R = A \times B$ must satisfy $\mathrm{disc}(R) \geq 2\epsilon$.

This implies $\frac{|A|\cdot|B|}{2^{2n}} \geq 2\epsilon$, so $|A| \cdot |B| \geq 2\epsilon \cdot 2^{2n}$.

By the AM-GM inequality, $|A| \cdot |B| \leq 2^{2n}$ with equality when $|A| = |B| = 2^n$. The constraint $|A| \cdot |B| \geq 2\epsilon \cdot 2^{2n}$ combined with the partition property implies that the total number of rectangles satisfies:

$$2^c \geq \frac{2^{2n}}{|A| \cdot |B|} \geq \frac{2^{2n}}{2^{2n}/2\epsilon} = 2\epsilon \cdot 2^{2n}/2^n = 2\epsilon \cdot 2^n$$

Taking logarithms: $c \geq n + \log(2\epsilon) = n - \log(1/\epsilon) + 1$.

Therefore, $R_{\frac{1}{2}-\epsilon}(\text{IP}) \geq n - O(\log(1/\epsilon))$. □

E.12.4 Gemini

Proof. By Yao's Minimax Principle, the randomized communication complexity $R^{\text{pub}}_\delta(f)$ is lower-bounded by the distributional complexity $D^\mu_\delta(f)$ for any distribution μ. We consider the uniform distribution μ over $\{0,1\}^n \times \{0,1\}^n$. Let the allowed error probability be $\delta = \frac{1}{2} - \epsilon$.

Let Π be a deterministic protocol for IP with distributional error at most δ under μ. Let C be the communication cost of Π. The protocol partitions the input space into combinatorial rectangles $\mathcal{R} = \{R_1, \ldots, R_k\}$ with $k \leq 2^C$. For each rectangle R_j, the protocol outputs a constant bit $v_j \in \{0,1\}$.

The correlation (advantage) of the protocol with the function IP is given by:

$$\text{Adv}(\Pi) = \sum_{(x,y)\in\{0,1\}^{2n}} \mu(x,y)(-1)^{\Pi(x,y)\oplus \text{IP}(x,y)}.$$

Since the error probability is at most $\frac{1}{2} - \epsilon$, the probability of correctness is at least $\frac{1}{2} + \epsilon$. Thus, the advantage satisfies:

$$|\text{Adv}(\Pi)| = |\Pr[\Pi = \text{IP}] - \Pr[\Pi \neq \text{IP}]| \geq \left(\frac{1}{2} + \epsilon\right) - \left(\frac{1}{2} - \epsilon\right) = 2\epsilon.$$

Substituting the uniform distribution $\mu(x,y) = 2^{-2n}$:

$$2\epsilon \leq \frac{1}{2^{2n}} \left| \sum_{j=1}^{k} \sum_{(x,y)\in R_j} (-1)^{v_j \oplus \text{IP}(x,y)} \right| \leq \frac{1}{2^{2n}} \sum_{j=1}^{k} \left| \sum_{(x,y)\in R_j} (-1)^{\text{IP}(x,y)} \right|.$$

Let H be the communication matrix of IP, where $H_{x,y} = (-1)^{\text{IP}(x,y)}$. This is the Hadamard matrix of dimension $N \times N$ where $N = 2^n$. The inner sum corresponds to $\mathbf{1}_{X_j}^T H \mathbf{1}_{Y_j}$ where $R_j = X_j \times Y_j$. The spectral norm (largest singular value) of the Hadamard matrix H is $\|H\| = \sqrt{N \cdot N}/\sqrt{N}$? No, $HH^T = NI$, so $\|H\| = \sqrt{N} = 2^{n/2}$. Using the spectral bound bound for the discrepancy of a rectangle:

$$\left| \sum_{(x,y)\in R_j} (-1)^{\text{IP}(x,y)} \right| \leq \sqrt{|X_j|} \cdot \|H\| \cdot \sqrt{|Y_j|} = \sqrt{|R_j|} 2^{n/2}.$$

Substituting this back into the advantage inequality:

$$2\epsilon \cdot 2^{2n} \leq \sum_{j=1}^{k} \sqrt{|R_j|} 2^{n/2}.$$

Dividing by $2^{n/2}$:

$$2\epsilon 2^{1.5n} \leq \sum_{j=1}^{k} \sqrt{|R_j|}.$$

Applying the Cauchy-Schwarz inequality to the sum:

$$\sum_{j=1}^{k} \sqrt{|R_j|} \leq \sqrt{\sum_{j=1}^{k} 1} \cdot \sqrt{\sum_{j=1}^{k} |R_j|} = \sqrt{k} \cdot \sqrt{2^{2n}} = \sqrt{k} 2^n.$$

Combining the inequalities:

$$2\epsilon 2^{1.5n} \leq \sqrt{k}2^n \implies \sqrt{k} \geq 2\epsilon 2^{n/2}.$$

Squaring both sides:

$$k \geq 4\epsilon^2 2^n.$$

Taking the logarithm to bound the communication cost $C \geq \log_2 k$:

$$C \geq \log_2(4\epsilon^2 2^n) = 2 + 2\log_2 \epsilon + n = n - 2\log_2(1/\epsilon) + 2.$$

Therefore, $C \geq n - O(\log(1/\epsilon))$. Since this holds for any deterministic protocol with error $\frac{1}{2} - \epsilon$ under the uniform distribution, we conclude:

$$R_{\frac{1}{2}-\epsilon}(\mathrm{IP}) \geq n - O(\log(1/\epsilon)).$$

□

E.12.5 Grok

Proof. Let $\mathrm{IP} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be defined by $\mathrm{IP}(x,y) = \sum_{i=1}^n x_i y_i \pmod 2$. Let μ be the uniform distribution over $\{0,1\}^n \times \{0,1\}^n$. The discrepancy of IP with respect to μ is defined as

$$textdisc_\mu(\mathrm{IP}) = \max_R \left| \mathop{\mathbb{E}}_{(x,y)\sim\mu} \left[(-1)^{\mathrm{IP}(x,y)} \cdot \mathbf{1}_{(x,y)\in R} \right] \right|,$$

where the maximum is taken over all combinatorial rectangles $R = A \times B$ with $A, B \subseteq \{0,1\}^n$.

We claim that $textdisc_\mu(\mathrm{IP}) = 2^{-n}$. To see this, note that

$$\mathop{\mathbb{E}}_{(x,y)\sim\mu} \left[(-1)^{\mathrm{IP}(x,y)} \cdot \mathbf{1}_{(x,y)\in R} \right] = 2^{-2n} \sum_{x\in A} \sum_{y\in B} (-1)^{\sum_i x_i y_i}.$$

The inner double sum equals

$$\sum_{y\in B} \sum_{x\in A} \prod_{i=1}^n (-1)^{x_i y_i} = \sum_{y\in B} \prod_{i=1}^n \sum_{x_i=0}^1 \mathbf{1}_{x_i\in A_i} (-1)^{x_i y_i},$$

where $A_i = \{x_i \mid x \in A\}$, but it is more straightforward to compute

$$\sum_x \mathbf{1}_{x\in A} (-1)^{\sum_i x_i y_i} = \sum_{x\in A} \prod_i (-1)^{x_i y_i}.$$

Since the variables are independent,

$$\sum_{x\in\{0,1\}^n} (-1)^{\sum_i x_i y_i} = \prod_i \sum_{x_i=0}^1 (-1)^{x_i y_i}.$$

For each i, if $y_i = 0$ then $\sum_{x_i} 1 = 2$; if $y_i = 1$ then $\sum_{x_i} (-1)^{x_i} = 1 - 1 = 0$. Thus, the full sum over x is 2^n if $y = 0^n$ and 0 otherwise. Therefore,

$$\sum_{x\in A} (-1)^{\sum_i x_i y_i} = \mathbf{1}_{y=0^n} \cdot |A|,$$

since for $y = 0^n$, $(-1)^0 = 1$ for each x. Hence,

$$\sum_{y\in B} \sum_{x\in A} (-1)^{\mathrm{IP}(x,y)} = |A| \cdot \mathbf{1}_{0^n\in B}.$$

It follows that the absolute value is at most $|A| \le 2^n$, achieved when $A = \{0,1\}^n$ and $0^n \in B$. Thus,

$$textdisc_\mu(\mathrm{IP}) = 2^n/2^{2n} = 2^{-n}.$$

Now, suppose there is a private-coin randomized protocol Π for IP with worst-case error at most $1/2-\epsilon$ and communication cost c. By Yao's principle, the private-coin randomized complexity $R^{\mathrm{priv}}_{1/2-\epsilon}(\mathrm{IP})$ equals $\max_\nu D^\nu_{1/2-\epsilon}(\mathrm{IP})$, where the maximum is over all distributions ν and D^ν_δ is the minimal cost of a deterministic protocol with error at most δ under ν. Thus, $D^\mu_{1/2-\epsilon}(\mathrm{IP}) \le R^{\mathrm{priv}}_{1/2-\epsilon}(\mathrm{IP}) \le c$. For a deterministic protocol with communication c and average error at most $1/2-\epsilon$ under μ, the correlation satisfies

$$\mathop{\mathbb{E}}_{\mu}[(2\Pi(x,y)-1)(2\mathrm{IP}(x,y)-1)] = 2\Pr_{\mu}[\Pi(x,y)=\mathrm{IP}(x,y)] - 1 \ge 2\epsilon,$$

since the average success probability is at least $1/2+\epsilon$. A deterministic protocol with c bits of communication corresponds to a partition of the input space into at most 2^c monochromatic rectangles. Let $h(x,y) = 2\Pi(x,y)-1 \in \{\pm 1\}$. Then h can be written as a sum $h(x,y) = \sum_{j=1}^k s_j \mathbf{1}_{(x,y)\in R_j}$ where $k \le 2^c$, $s_j = \pm 1$, and the R_j are disjoint rectangles (though the disjointness is not necessary for the bound). Thus,

$$\left|\mathop{\mathbb{E}}_{\mu}[(-1)^{\mathrm{IP}(x,y)}h(x,y)]\right| = \left|\sum_j s_j \mathop{\mathbb{E}}_{\mu}[(-1)^{\mathrm{IP}(x,y)}\mathbf{1}_{R_j}]\right| \le \sum_j \left|\mathop{\mathbb{E}}_{\mu}[(-1)^{\mathrm{IP}(x,y)}\mathbf{1}_{R_j}]\right| \le 2^c \cdot textdisc_\mu(\mathrm{IP}) = 2^{c-n}.$$

Note that $(2\mathrm{IP}-1) = -(-1)^{\mathrm{IP}(x,y)}$, since if $\mathrm{IP}=0$, $2\cdot 0-1=-1$, $(-1)^0=1$, $-1=-1$; if $\mathrm{IP}=1$, $1=-(-1)$. Wait, $(-1)^{\mathrm{IP}}=1$ if 0, -1 if 1; so $-(-1)^{\mathrm{IP}} = -1$ if 0, 1 if 1, which is $2\mathrm{IP}-1$. Yes. Thus,

$$\mathbb{E}[(2\mathrm{IP}-1)h] = \mathbb{E}[-(-1)^{\mathrm{IP}}h] = -\mathbb{E}[(-1)^{\mathrm{IP}}h],$$

so

$$|\mathbb{E}[(2\mathrm{IP}-1)h]| = \left|\mathbb{E}[(-1)^{\mathrm{IP}}h]\right| \le 2^{c-n}.$$

Therefore, $2\epsilon \le 2^{c-n}$, so $2^{c-n} \ge 2\epsilon$, hence $c-n \ge \log_2(2\epsilon) = 1+\log_2\epsilon$, or $c \ge n+1+\log_2\epsilon = n-\log_2(1/\epsilon)+1-\log_2 1 = n - O(\log(1/\epsilon))$.

This gives $R^{\mathrm{priv}}_{1/2-\epsilon}(\mathrm{IP}) \ge n - O(\log(1/\epsilon))$. By Newman's theorem, any public-coin protocol with error δ can be simulated by a private-coin protocol with error $\delta + o(1)$ and additional communication $O(\log n + \log(1/\delta))$. Adjusting constants, since $\delta = 1/2-\epsilon$, $\log(1/\delta) = O(1)$, and for the advantage 2ϵ, the simulation adds $O(\log n + \log(1/\epsilon))$. Thus, $R_{1/2-\epsilon}(\mathrm{IP}) \ge R^{\mathrm{priv}}_{1/2-\epsilon}(\mathrm{IP}) - O(\log n + \log(1/\epsilon)) \ge n - O(\log n + \log(1/\epsilon))$. However, in the asymptotic sense for the theorem statement, which omits the $\log n$ term, the bound holds as stated, absorbing constants appropriately. □

E.13 Problem 13

E.13.1 Problem Statement

Theorem E.13. *For "most" Boolean functions $D^{uniform}_\varepsilon(f) = n - O(\log \frac{1}{\varepsilon})$. That is, only an exponentially small fraction of the functions have better complexity. Hint: Pick a function at random and compute the probability that it has a "large," "almost monochromatic" rectangle.*

E.13.2 GPT

Proof. Fix $n \in \mathbb{N}$ and $\varepsilon \in (0, 1/4]$. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be chosen uniformly at random from all Boolean functions on $2n$ bits, i.e., the values $\{f(x,y)\}_{(x,y)\in\{0,1\}^{2n}}$ are i.i.d. unbiased bits. We consider deterministic protocols under the uniform input distribution and define $D^{\mathrm{uniform}}_\varepsilon(f)$ to be the minimum cost c of a deterministic protocol Π such that

$$\Pr_{(X,Y)\sim \mathrm{Unif}(\{0,1\}^{2n})}\big[\Pi(X,Y) \ne f(X,Y)\big] \le \varepsilon.$$

We show that with probability at least $1 - 2^{-\Omega(2^n)}$ over f we have $D^{\mathrm{uniform}}_{\varepsilon}(f) \geq n - C\log(1/\varepsilon)$ for a universal constant $C > 0$, which implies the claimed statement.

Let Π be any deterministic communication protocol of cost c. For each transcript t of Π, the set of inputs leading to t is a combinatorial rectangle $R_t = A_t \times B_t \subseteq \{0,1\}^n \times \{0,1\}^n$, and Π outputs a fixed bit $b_t \in \{0,1\}$ on all inputs in R_t. The rectangles $\{R_t\}_t$ form a partition of $\{0,1\}^{2n}$, and there are at most 2^c transcripts. Under the uniform distribution, define $\mu(S) = |S|/2^{2n}$ for $S \subseteq \{0,1\}^{2n}$. Then $\sum_t \mu(R_t) = 1$. Hence there exists a transcript $t^\star$ such that $\mu(R_{t^\star}) \geq 2^{-c}$.

For any rectangle $R \subseteq \{0,1\}^{2n}$ and any bit $b \in \{0,1\}$, define the b-error of f on R as

$$\mathrm{err}_b(f; R) = \frac{1}{|R|}\big|\{(x,y) \in R : f(x,y) \neq b\}\big|.$$

If Π has uniform error at most ε, then

$$\varepsilon \geq \sum_t \mu(R_t)\,\mathrm{err}_{b_t}(f; R_t).$$

Since all summands are nonnegative, we have $\mathrm{err}_{b_{t^\star}}(f; R_{t^\star}) \leq \varepsilon/\mu(R_{t^\star}) \leq \varepsilon\, 2^c$. Therefore, if $c \leq n - \alpha$ and α is such that $\varepsilon\, 2^c \leq 2^{-1}$, then there exists a rectangle R with $\mu(R) \geq 2^{-c}$ and $\mathrm{err}_b(f; R) \leq \delta$ for some b, where $\delta := \varepsilon\, 2^c$.

We next bound the probability (over random f) that there exists a rectangle R of measure at least 2^{-c} on which f is δ-close to constant. Fix a rectangle R with $|R| = m$. For random f, the multiset $\{f(z) : z \in R\}$ consists of m i.i.d. unbiased bits. For a fixed $b \in \{0,1\}$, the number of disagreements with b is $\mathrm{Bin}(m, 1/2)$. By the standard Chernoff bound, for any $\delta \in (0, 1/2)$,

$$\Pr\big[\mathrm{err}_b(f; R) \leq \delta\big] = \Pr\big[\mathrm{Bin}(m,1/2) \leq \delta m\big] \leq \exp\big(-2(1/2 - \delta)^2 m\big) \leq \exp(-m/8),$$

where the last inequality uses $\delta \leq 1/4$ so that $(1/2 - \delta)^2 \geq 1/16$. Thus

$$\Pr\big[\exists\, b \in \{0,1\} : \mathrm{err}_b(f; R) \leq \delta\big] \leq 2e^{-m/8}.$$

Now we union bound over all rectangles $R = A \times B$ with $|R| \geq M$, where $M := 2^{2n-c}$ corresponds to $\mu(R) \geq 2^{-c}$. The number of such rectangles is at most the total number of rectangles, namely $2^{2^n} \cdot 2^{2^n} = 2^{2^{n+1}}$, since there are 2^{2^n} choices for each of $A, B \subseteq \{0,1\}^n$. Therefore,

$$\Pr\big[\exists\, R \text{ rectangle with } |R| \geq M \text{ and } \exists b : \mathrm{err}_b(f; R) \leq \delta\big] \leq 2^{2^{n+1}+1}\exp(-M/8).$$

If $c \leq n - \alpha$ then $M = 2^{2n-c} \geq 2^{n+\alpha}$. Hence

$$2^{2^{n+1}+1}\exp(-M/8) \leq \exp\big((2^{n+1}+1)\ln 2 - 2^{n+\alpha}/8\big).$$

Choose $\alpha \geq 6$ so that $2^{n+\alpha}/8 \geq 2^{n+2}\ln 2$ for all $n \geq 1$, which holds since it is equivalent to $2^{\alpha-3} \geq 4\ln 2$. Then the exponent is at most -2^{n+1} for all n sufficiently large, and in any case it is $-\Omega(2^n)$. Thus for such α,

$$\Pr\big[\exists\, R \text{ with } \mu(R) \geq 2^{-c} \text{ and } \exists b : \mathrm{err}_b(f; R) \leq \delta\big] \leq 2^{-\Omega(2^n)}.$$

We now instantiate parameters. Let $c := n - \alpha$ and set $\delta := \varepsilon\, 2^c = \varepsilon\, 2^{n-\alpha}$. Choose $\alpha := \lceil \log_2(1/\varepsilon) \rceil + 3$, so that $\delta \leq 1/8 \leq 1/4$ and the Chernoff bound above applies. Also $\alpha \geq 6$ whenever $\varepsilon \leq 2^{-3}$; if $\varepsilon > 2^{-3}$ then $\log_2(1/\varepsilon) = O(1)$ and the final bound remains valid after adjusting constants, so we may assume $\alpha \geq 6$ without loss of generality by increasing the implicit constant in $O(\log(1/\varepsilon))$.

Suppose there exists a deterministic protocol Π of cost at most c with uniform error at most ε. As shown earlier, there exists a rectangle R with $\mu(R) \geq 2^{-c}$ and a bit b such that $\mathrm{err}_b(f; R) \leq \delta$. Therefore, for a random f, the event $\{D^{\mathrm{uniform}}_{\varepsilon}(f) \leq c\}$ implies the event that there exists such a large almost monochromatic rectangle. Consequently,

$$\Pr_f\big[D^{\mathrm{uniform}}_{\varepsilon}(f) \leq n - \alpha\big] \leq 2^{-\Omega(2^n)}.$$

Equivalently, with probability at least $1 - 2^{-\Omega(2^n)}$ over f we have $D^{\mathrm{uniform}}_{\varepsilon}(f) \geq n - \alpha = n - O(\log(1/\varepsilon))$. This shows that all but an exponentially small fraction of Boolean functions satisfy $D^{\mathrm{uniform}}_{\varepsilon}(f) = n - O(\log(1/\varepsilon))$, and hence only an exponentially small fraction have better complexity. $\square$

E.13.3 Claude

Proof. We prove that for a randomly chosen Boolean function $f : \{0,1\}^n \to \{0,1\}$, with high probability, $D_\varepsilon^{\text{uniform}}(f) \geq n - O(\log \frac{1}{\varepsilon})$.

Let f be chosen uniformly at random from all 2^{2^n} Boolean functions on n variables. We will show that any protocol with depth $d < n - O(\log \frac{1}{\varepsilon})$ fails to compute f with error at most ε with high probability.

Consider a deterministic protocol π of depth d. This protocol partitions $\{0,1\}^n \times \{0,1\}^n$ into at most 2^d monochromatic rectangles $R_1, \ldots, R_k$ where $k \leq 2^d$. For the protocol to compute f with error at most ε under the uniform distribution, we need

$$\Pr_{(x,y)\sim\text{Unif}}[f(x \oplus y) \neq \pi(x,y)] \leq \varepsilon.$$

For a rectangle R_i with output value $b_i \in \{0,1\}$, define the error on R_i as

$$\text{err}(R_i) = \frac{1}{2^{2n}}|\{(x,y) \in R_i : f(x \oplus y) \neq b_i\}|.$$

The total error is $\sum_{i=1}^k \text{err}(R_i)$. For the protocol to succeed, we need $\sum_{i=1}^k \text{err}(R_i) \leq \varepsilon$.

Let R be a rectangle of size $|R| = s \cdot 2^{2n}$ for some $s \in (0,1]$. The inputs $(x,y) \in R$ correspond to at most $s \cdot 2^{2n}$ pairs, which map to at most 2^n distinct values $z = x \oplus y$.

For a random function f, each value $f(z)$ is chosen independently and uniformly from $\{0,1\}$. Fix a rectangle R and output value b. The probability that R has error at most αs is bounded by

$$\Pr[\text{err}(R) \leq \alpha s] \leq \Pr\left[\text{Binomial}(s \cdot 2^{2n}, 1/2) \leq \alpha s \cdot 2^{2n}\right].$$

By Chernoff bounds, for $\alpha < 1/2$, this probability is at most $\exp(-\Omega(s \cdot 2^{2n}))$.

Now consider a protocol with $d = n - c\log(1/\varepsilon)$ for sufficiently large constant c. The protocol has at most $2^d = 2^n/\varepsilon^c$ rectangles.

For each rectangle R_i of size $s_i \cdot 2^{2n}$, we have $\sum_{i=1}^k s_i \leq 1$. For the protocol to achieve error ε, at least one rectangle must have $\text{err}(R_i) \leq \varepsilon/k$.

If $s_i \geq \varepsilon^{c+1}$, then the probability that $\text{err}(R_i) \leq \varepsilon/k \leq \varepsilon^{c+1}$ is at most $\exp(-\Omega(\varepsilon^{c+1} \cdot 2^{2n}))$.

By union bound over all possible protocols of depth d and all rectangles, the total number of such protocols is at most $2^{O(d \cdot 2^n)} = 2^{O(2^n/\varepsilon^c)}$.

The probability that any such protocol succeeds is at most

$$2^{O(2^n/\varepsilon^c)} \cdot 2^d \cdot \exp(-\Omega(\varepsilon^{c+1} \cdot 2^{2n})) = o(1)$$

for sufficiently large c and n.

Therefore, with probability $1 - o(1)$, a random Boolean function f satisfies $D_\varepsilon^{\text{uniform}}(f) \geq n - O(\log \frac{1}{\varepsilon})$. □

E.13.4 Gemini

Proof. Let $N = 2^n$. The total number of inputs is $N^2 = 2^{2n}$. We consider the uniform distribution over the inputs. Let ε denote the advantage (bias) of the protocol, meaning the error probability is at most $\frac{1}{2} - \varepsilon$. A deterministic protocol with communication complexity c partitions the input space $X \times Y$ into at most 2^c disjoint combinatorial rectangles $R_1, \ldots, R_k$, where $k \leq 2^c$. For the protocol to approximate a function f with advantage ε, it must correctly classify at least a $(\frac{1}{2} + \varepsilon)$ fraction of the inputs. This implies that there exists a partition of the matrix of f into 2^c rectangles such that the function f has a bias of at least ε on the union of these rectangles (weighted by size). Specifically, this implies the existence of at least one "large" rectangle R on which f is "almost monochromatic" (biased).

We bound the probability that a random Boolean function f admits such a protocol. Let $c = n - \Delta$. The average size of a rectangle in a partition of 2^{2n} inputs into 2^c sets is $S_{avg} = 2^{2n}/2^c = 2^{2n-(n-\Delta)} = 2^{n+\Delta}$. Consider a fixed rectangle R of size S. For a random function f, the values on R are independent uniform bits. The discrepancy of f on R, denoted $\Delta(f,R) = |\sum_{(x,y)\in R}(-1)^{f(x,y)}|$, behaves like the displacement

of a random walk of length S. By the Chernoff bound, the probability that f has advantage ε on R (i.e., agrees with a constant bit on $\geq (\frac{1}{2} + \varepsilon)S$ inputs) is bounded by:

$$\Pr[\text{Advantage on } R \geq \varepsilon] \leq 2e^{-2\varepsilon^2 S}.$$

The number of possible combinatorial rectangles in an $N \times N$ matrix is $(2^N)^2 = 2^{2N} = 2^{2\cdot 2^n}$. By the union bound, the probability that *there exists* any rectangle of size S with advantage ε is at most:

$$P_{exists} \leq 2^{2\cdot 2^n} \cdot 2e^{-2\varepsilon^2 S}.$$

We require this probability to be small (e.g., < 1) to show that most functions do not have such a rectangle. Taking logarithms:

$$2 \cdot 2^n - 2\varepsilon^2 S \log_2 e < 0 \implies 2^{n+1} < c_0 \varepsilon^2 S,$$

where $c_0 = 2\log_2 e$. Substituting $S = 2^{n+\Delta}$:

$$2^{n+1} < c_0 \varepsilon^2 2^{n+\Delta} \implies 2^1 < c_0 \varepsilon^2 2^{\Delta}.$$

Solving for Δ:

$$2^{\Delta} > \frac{2}{c_0 \varepsilon^2} \implies \Delta > \log_2\left(\frac{1}{\varepsilon^2}\right) - O(1) = 2\log_2 \frac{1}{\varepsilon} - O(1).$$

Thus, for a protocol to exist with high probability, we would need Δ to be small. Conversely, if $\Delta \approx 2\log(1/\varepsilon)$, such biased rectangles do not exist for most functions. This implies that for most functions, the communication complexity D must satisfy:

$$D \geq n - \Delta \approx n - O\left(\log \frac{1}{\varepsilon}\right).$$

Since $D(f) \leq n + 1$ for any function, we have $D_{\varepsilon}^{\text{uniform}}(f) = n - O(\log \frac{1}{\varepsilon})$. □

E.13.5 Grok

Proof. Consider the Boolean function $f : \{0,1\}^n \to \{0,1\}$. The uniform ε-error deterministic communication complexity $D_{\varepsilon}^{\text{uniform}}(f)$ is the minimal depth of a deterministic protocol tree where, for every input $(x, y) \in \{0,1\}^n \times \{0,1\}^n$, the leaf reached outputs $f(x, y)$ with probability at least $1 - \varepsilon$ over the uniform distribution on inputs.

Equivalently, $D_{\varepsilon}^{\text{uniform}}(f)$ is the minimal number of bits communicated in a deterministic protocol that computes f correctly on at least a $(1 - \varepsilon)$-fraction of inputs under the uniform measure μ on $\{0,1\}^{2n}$.

A protocol of depth d partitions the input matrix $M_f \in \{0,1\}^{2^n \times 2^n}$ into at most 2^d monochromatic rectangles (leaves). If the protocol is correct on a $(1 - \varepsilon)$-fraction of inputs, then the total measure of inputs in non-monochromatic leaves is at most ε, so the union of the monochromatic rectangles covers at least measure $1 - \varepsilon$.

Thus, $D_{\varepsilon}^{\text{uniform}}(f) \leq d$ if and only if there exists a collection of at most 2^d monochromatic rectangles whose total measure is at least $1 - \varepsilon$.

To show that for most f, $D_{\varepsilon}^{\text{uniform}}(f) = n - O(\log(1/\varepsilon))$, we prove that with probability $1 - 2^{-\Omega(2^n \varepsilon^2)}$ over the random choice of f, every collection of $2^{n - c\log(1/\varepsilon)}$ monochromatic rectangles has total measure at most $1 - \varepsilon$, for a suitable constant $c > 0$. This implies that $D_{\varepsilon}^{\text{uniform}}(f) > n - c\log(1/\varepsilon)$.

Fix $\delta = \varepsilon/2$ and $k = n - c\log(1/\delta)$ with c to be chosen later. We show that the probability that there exists a collection of 2^k monochromatic rectangles covering measure at least $1 - \delta$ is at most $2^{-\Omega(2^n \delta^2)}$.

A rectangle $R = A \times B \subseteq \{0,1\}^n \times \{0,1\}^n$, with $A, B \neq \emptyset$, is monochromatic for a random f if f is constant on R. The measure of R is $\mu(R) = |A|/2^n \cdot |B|/2^n$.

For a fixed rectangle $R = A \times B$ with $|A| = a \cdot 2^n$, $|B| = b \cdot 2^n$ where $a, b > 0$, the probability that f is constant on R is $2 \cdot 2^{-ab \cdot 2^{2n}}$ (two choices for the constant value, and each of the $ab \cdot 2^{2n}$ entries must match it).

We call a rectangle R large if $\mu(R) \geq \delta^2/4$. For such a large rectangle, the probability that it is monochromatic is

$$\Pr[R \text{ monochromatic}] = 2 \cdot 2^{-ab \cdot 2^{2n}} \leq 2 \cdot 2^{-\delta^2 \cdot 2^{2n}/4} = 2^{1-\delta^2 2^{2n}/4}.$$

The number of possible rectangles is at most $(2^{2^n})^2 = 2^{2 \cdot 2^n}$. Thus, the expected number of large monochromatic rectangles is at most

$$2^{2 \cdot 2^n} \cdot 2^{1-\delta^2 2^{2n}/4} = 2^{2 \cdot 2^n + 1 - \delta^2 2^{2n}/4}.$$

Choosing c sufficiently large ensures that $\delta^2 2^{2n}/4 - 2 \cdot 2^n - 1 \geq \Omega(2^n \delta^2)$, so the expected number is $2^{-\Omega(2^n \delta^2)}$. By Markov's inequality, the probability that there exists even one large monochromatic rectangle is at most $2^{-\Omega(2^n \delta^2)}$.

Now suppose there is no large monochromatic rectangle. Consider any collection $\mathcal{R}$ of 2^k monochromatic rectangles. Since each has measure $\mu(R_i) < \delta^2/4$, the total measure covered is

$$\mu\left(\bigcup_{R_i \in \mathcal{R}} R_i\right) \leq \sum_{i=1}^{2^k} \mu(R_i) < 2^k \cdot \frac{\delta^2}{4}.$$

With $k = n - c\log(1/\delta)$ and $c \geq 3$ (say), we have $2^k = 2^n/\delta^{O(1)} \ll 4/\delta$ for small δ, so

$$2^k \cdot \frac{\delta^2}{4} < \delta = \varepsilon/2 < 1 - \varepsilon$$

for sufficiently small ε. Thus, no such collection covers measure $1 - \varepsilon$.

The probability that there exists a large monochromatic rectangle is exponentially small, $2^{-\Omega(2^n \varepsilon^2)}$. Therefore, with probability $1 - 2^{-\Omega(2^n \varepsilon^2)}$ over random f, there is no large monochromatic rectangle, and hence every collection of $2^{n - O(\log(1/\varepsilon))}$ monochromatic rectangles covers measure less than $1 - \varepsilon$. This implies $D_\varepsilon^{\text{uniform}}(f) > n - O(\log(1/\varepsilon))$.

Since the total number of Boolean functions is $2^{2^{2n}}$, the fraction of functions with $D_\varepsilon^{\text{uniform}}(f) \leq n - O(\log(1/\varepsilon))$ is at most $2^{-\Omega(2^n \varepsilon^2)}$, which is exponentially small in 2^n. Hence, for all but an exponentially small fraction of Boolean functions, $D_\varepsilon^{\text{uniform}}(f) = n - O(\log(1/\varepsilon))$. □

E.14 Problem 14

E.14.1 Problem Statement

Theorem E.14 (Exercise 3.32)**.** *Let* $\mathrm{Disc}(f) = \min_r \mathrm{Disc}_r(f)$.

(a) $\mathrm{Disc}(\mathrm{DISJ}) \geq \frac{1}{2n+1}$.

(b) *For every function* f, $R^{pub}(f) \leq \left(\frac{1}{\mathrm{Disc}(f)}\right)^{O(1)}$. *(Hint: Use the distributional complexity.)*

E.14.2 GPT

Proof. **(a)** We begin by considering the definition of $\mathrm{Disc}(f)$:

$$\mathrm{Disc}(f) = \min_r \mathrm{Disc}_r(f),$$

where $\mathrm{Disc}_r(f)$ denotes the distributional complexity of f with respect to a radius r. The goal is to prove that

$$\mathrm{Disc}(\mathrm{DISJ}) \geq \frac{1}{2n+1}.$$

For the function DISJ, the distributional complexity is related to the minimum radius for which a decision tree can distinguish between the two distributions. A well-known result for the distributional complexity of the disjointness function DISJ is that the complexity is lower-bounded by $\frac{1}{2n+1}$, where n is the number of bits involved in the disjointness problem. This lower bound can be derived from the fact that for

disjointness, the decision tree complexity is at least proportional to the number of bits in the input, and thus we conclude that

$$\mathrm{Disc}(\mathrm{DISJ}) \geq \frac{1}{2n+1}.$$

(b) To prove the second part of the theorem, we use the hint about distributional complexity. Specifically, we are tasked with showing that for any function f,

$$R^{pub}(f) \leq \left(\frac{1}{\mathrm{Disc}(f)}\right)^{O(1)}.$$

Recall that $R^{pub}(f)$ denotes the public communication complexity of f, and the distributional complexity $\mathrm{Disc}(f)$ quantifies the difficulty of distinguishing the two possible distributions that define f. A standard result in communication complexity is that the public communication complexity is related to the distributional complexity by the following inequality:

$$R^{pub}(f) \leq \left(\frac{1}{\mathrm{Disc}(f)}\right)^{O(1)}.$$

This bound arises from the fact that any public communication protocol can be simulated by a distributional complexity argument, and the constant in the $O(1)$ notation depends on the specific function f and the way the communication complexity is derived from the distributional complexity. Therefore, we conclude that

$$R^{pub}(f) \leq \left(\frac{1}{\mathrm{Disc}(f)}\right)^{O(1)}.$$

□

E.14.3 Claude

Proof. We prove both parts of the theorem.

Part (a): We show that $\mathrm{Disc}(\mathrm{DISJ}) \geq \frac{1}{2n+1}$.

Recall that $\mathrm{DISJ}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is defined by $\mathrm{DISJ}_n(x,y) = 1$ if and only if $x \wedge y = 0^n$ (i.e., the sets are disjoint).

Let μ be the uniform distribution over disjoint pairs, that is, μ is supported on pairs (x,y) where $x \wedge y = 0^n$. For any rectangle $R = A \times B$, we have

$$\mu(R) = \frac{|A \times B \cap \mathrm{DISJ}^{-1}(1)|}{|\mathrm{DISJ}^{-1}(1)|}.$$

Consider any rectangle $R = A \times B$ that is monochromatic for DISJ. If R is a 1-rectangle (all pairs in R are disjoint), then for any $x \in A$ and $y \in B$, we have $x \wedge y = 0^n$.

Fix any $x \in A$. The number of $y \in \{0,1\}^n$ such that $x \wedge y = 0^n$ is exactly $2^{n-|x|}$ where $|x|$ denotes the Hamming weight of x. Since B must be contained in this set, we have $|B| \leq 2^{n-|x|}$.

Similarly, for the uniform distribution ν over all pairs $(x,y) \in \{0,1\}^n \times \{0,1\}^n$, we have

$$\nu(R) = \frac{|A| \cdot |B|}{2^{2n}}.$$

The probability that a uniformly random pair is disjoint is $\mathbb{P}_{(x,y)\sim\nu}[\mathrm{DISJ}(x,y)=1] = \sum_{k=0}^{n} \binom{n}{k} \frac{1}{2^n} \cdot \frac{1}{2^k} = \frac{1}{2^n} \sum_{k=0}^{n} \binom{n}{k} \frac{1}{2^k} = \frac{1}{2^n} \cdot \left(\frac{3}{2}\right)^n = \left(\frac{3}{2n}\right)^n \cdot \frac{2^n}{2^n} = \frac{3^n}{2^{2n}}$.

By a standard argument using the hard distribution μ being the uniform distribution over disjoint pairs, and applying the probabilistic method with Razborov's technique, one can show that for any distribution π and any 1-rectangle R,

$$\frac{\mu(R)}{\nu(R)} \leq 2n+1.$$

Therefore, $\mathrm{Disc}_\mu(\mathrm{DISJ}) \geq \frac{1}{2n+1}$, which implies $\mathrm{Disc}(\mathrm{DISJ}) \geq \frac{1}{2n+1}$.

Part (b): We show that for every function f, $R^{pub}(f) \leq \left(\frac{1}{\mathrm{Disc}(f)}\right)^{O(1)}$.

Let μ be the distribution achieving $\mathrm{Disc}(f) = \mathrm{Disc}_\mu(f)$. By definition of discrepancy, every rectangle R that is monochromatic for f satisfies

$$\left|\mu(R \cap f^{-1}(b)) - \nu(R \cap f^{-1}(b))\right| \leq \mathrm{Disc}(f)$$

where ν is the uniform distribution and $b \in \{0, 1\}$.

By the distributional complexity theorem, there exists a deterministic protocol Π that computes f correctly on all inputs in the support of μ with communication cost $D^\mu(f) = O(\log(1/\mathrm{Disc}(f)))$.

For public-coin randomized complexity, Alice and Bob can use shared randomness to sample from μ, and then use the protocol Π. By standard amplification and the connection between distributional and randomized complexity, we have

$$R^{pub}(f) \leq O(D^\mu(f)) = O(\log(1/\mathrm{Disc}(f))) = \left(\frac{1}{\mathrm{Disc}(f)}\right)^{O(1)}.$$

This completes the proof. □

E.14.4 Gemini

Proof. **Part (a):** We prove the bound for the Greater Than function, denoted here as DISJ following the problem statement (standard texts such as Kushilevitz and Nisan assign this property to GT_n). The function is defined by $\mathrm{DISJ}(x, y) = 1$ if $x > y$ (interpreted as integers) and 0 otherwise.

Let μ be an arbitrary probability distribution on $\{0, 1\}^n \times \{0, 1\}^n$. We construct a decomposition of the input space. For any input pair (x, y), let $i(x, y)$ be the first index where x and y differ. If $x = y$, we define $i(x, y) = n + 1$. We partition the input space into $2n + 1$ disjoint sets:

$$\begin{aligned} S_{k,0} &= \{(x, y) \mid i(x, y) = k \text{ and } x_k = 0, y_k = 1\} \quad \text{for } k \in \{1, \ldots, n\}, \\ S_{k,1} &= \{(x, y) \mid i(x, y) = k \text{ and } x_k = 1, y_k = 0\} \quad \text{for } k \in \{1, \ldots, n\}, \\ S_= &= \{(x, y) \mid x = y\}. \end{aligned}$$

Note that for $(x, y) \in S_{k,0}$, we have $x < y$, so $\mathrm{DISJ}(x, y) = 0$. For $(x, y) \in S_{k,1}$, we have $x > y$, so $\mathrm{DISJ}(x, y) = 1$. For $(x, y) \in S_=$, $\mathrm{DISJ}(x, y) = 0$. The sum of the measures of these $2n + 1$ disjoint sets is 1:

$$\mu(S_=) + \sum_{k=1}^{n} (\mu(S_{k,0}) + \mu(S_{k,1})) = 1.$$

By the Pigeonhole Principle, at least one of these sets must have measure at least $\frac{1}{2n+1}$.

Case 1: $\mu(S_=) \geq \frac{1}{2n+1}$. The set $S_=$ is the set of equal pairs. While $S_=$ is not a rectangle, we consider the contribution to the discrepancy. The function is constant 0 on $S_=$. It is a known property of the Greater Than function (via its spectral decomposition or explicit rectangle construction) that if the diagonal weight is large, there exists a monochromatic rectangle with sufficient weight, or the discrepancy is inherently bounded by the diagonal weight. Specifically, one can restrict μ to the diagonal to find a rectangle $R = \{x\} \times \{x\}$ (which is valid), but we need a general argument. A rigorous argument relies on the fact that GT_n has low nondeterministic communication complexity ($O(\log n)$), which implies its complement also has a small cover. The diagonal is covered by the complement.

Case 2: $\exists k$ such that $\mu(S_{k,1}) \geq \frac{1}{2n+1}$. The set $S_{k,1}$ is defined by the constraints: $x_j = y_j$ for $j < k$, $x_k = 1$, and $y_k = 0$. On this set, $\mathrm{DISJ}(x, y) = 1$. Thus, if $S_{k,1}$ were a rectangle, we would have discrepancy $\geq \frac{1}{2n+1}$. Although $S_{k,1}$ is not a rectangle, it is a sub-function equivalent to the Equality function on $k - 1$ bits restricted to the prefix. However, we can construct a rectangle $R = X' \times Y'$ contained in the high-probability region. A refined argument (from Nisan '93) shows that the spectral norm (and thus discrepancy) is bounded by these partition weights. Specifically, the discrepancy of f

is at least $\frac{1}{2\chi(f)}$, where $\chi(f)$ is the fractional chromatic number of the communication graph. For GT, $\chi(f) \leq 2n+1$. Thus:

$$\mathrm{Disc}_\mu(\mathrm{DISJ}) \geq \frac{1}{2n+1}.$$

Since this holds for any μ, $\mathrm{Disc}(\mathrm{DISJ}) \geq \frac{1}{2n+1}$.

Part (b): We prove that $R^{pub}(f) \leq \left(\frac{1}{\mathrm{Disc}(f)}\right)^{O(1)}$. By Yao's Minimax Principle, $R^{pub}(f) \approx \max_\mu D_\mu(f)$. It suffices to bound the distributional complexity $D_\mu(f)$ for an arbitrary distribution μ. Let $\Delta = \mathrm{Disc}_\mu(f)$. By definition, there exists a rectangle R such that $|\sum_{(x,y)\in R} \mu(x,y)(-1)^{f(x,y)}| \geq \Delta$. This implies that the rectangle R is a "weak predictor" for f with respect to μ. Specifically, the rectangle R has a correlation Δ with f. We can use a boosting algorithm (or the "rectangle covering" technique). A known theorem states that any function with discrepancy Δ has a randomized public-coin protocol with complexity:

$$R^{pub}(f) = O\left(\log^2\left(\frac{1}{\Delta}\right)\right).$$

Since $\log^2(1/\Delta) \leq (1/\Delta)^{O(1)}$ (in fact, it is polylogarithmic), the polynomial bound holds. More formally, we can construct a protocol that covers the support of the matrix using $O(1/\Delta)$ rectangles to achieve small error, or use the fact that the margin complexity is bounded by $1/\Delta$. Thus, $R^{pub}(f) \leq \mathrm{poly}(1/\mathrm{Disc}(f))$. □

E.14.5 Grok

Proof. We first prove part (a). Let $f = \mathrm{DISJ} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, where $f(x,y) = 1$ if there exists an $i \in [n]$ such that $x_i = y_i = 1$, and $f(x,y) = 0$ otherwise. For a probability distribution μ on $\{0,1\}^n \times \{0,1\}^n$, define $\mathrm{Disc}_\mu(f) = \max_R |\mu(R \cap f^{-1}(1)) - \mu(R \cap f^{-1}(0))|$, where the maximum is over all combinatorial rectangles R. Then $\mathrm{Disc}(f) = \min_\mu \mathrm{Disc}_\mu(f)$.

To show $\mathrm{Disc}(f) \geq \frac{1}{2n+1}$, we prove that for every μ, $\mathrm{Disc}_\mu(f) \geq \frac{1}{2n+1}$. Suppose, for contradiction, that there exists a μ such that $\mathrm{Disc}_\mu(f) < \frac{1}{2n+1}$. For each $i \in [n]$, let $R_i = \{(x,y) \mid x_i = y_i = 1\}$, which is a rectangle. On R_i, $f \equiv 1$, so $|\mu(R_i \cap f^{-1}(1)) - \mu(R_i \cap f^{-1}(0))| = \mu(R_i)$. Thus, $\mu(R_i) < \frac{1}{2n+1}$ for all i. Let E_i be the event $(x,y) \in R_i$. Then $\mu(f=0) = \mu(\cap_{i=1}^n \overline{E_i}) > 1 - \sum_{i=1}^n \mu(E_i) > 1 - \frac{n}{2n+1} = \frac{n+1}{2n+1}$, since $\sum \mu(E_i) < \frac{n}{2n+1}$.

For the full space $R = \{0,1\}^n \times \{0,1\}^n$, $|\mu(f=1) - \mu(f=0)| < \frac{1}{2n+1}$. Since $\mu(f=0) > \frac{1}{2}$, this implies $|1 - 2\mu(f=0)| = 2\mu(f=0) - 1 < \frac{1}{2n+1}$, so $\mu(f=0) < \frac{1+\frac{1}{2n+1}}{2} = \frac{n+1}{2n+1}$. This contradicts $\mu(f=0) > \frac{n+1}{2n+1}$. Thus, no such μ exists, and $\mathrm{Disc}(f) \geq \frac{1}{2n+1}$.

Now we prove part (b). Let $\delta = \mathrm{Disc}(f)$. We show that $R^{pub}_{1/3}(f) \leq O\left(\frac{1}{\delta}\right)$. By Newman's theorem, the private-coin randomized complexity satisfies $R^{priv}_{1/3}(f) \leq R^{pub}_{1/3}(f) + O(\log n)$, but since the bound is asymptotic, we focus on bounding the distributional complexity. By Yao's minimax principle, $R^{pub}_{1/3}(f) = \max_\mu D^\mu_{1/3}(f)$, up to constants.

For any fixed μ, since $\mathrm{Disc}(f) = \delta$, the induced function on any submatrix has $\mathrm{Disc} \geq \delta$, implying that for any distribution ν on any subset of the domain, $\mathrm{Disc}_\nu(f) \geq \delta$. Thus, $\mathrm{Disc}_\mu(f) \geq \delta$, so there exists a rectangle $R_1 = A_1 \times B_1$ with $|\mu(R_1 \cap f^{-1}(1)) - \mu(R_1 \cap f^{-1}(0))| \geq \delta$. Let $p_1 = \mu(R_1)$ and $a_1 = |2\Pr_\mu(f=1 \mid R_1) - 1| \geq \frac{\delta}{p_1}$, so the error if labeling R_1 by its majority is $\leq \frac{1}{2} - \frac{\delta}{2p_1}$. Since $a_1 \leq 1, p_1 \geq \delta$.

Let μ_1 be μ conditioned on the complement of R_1. Then $\mathrm{Disc}_{\mu_1}(f) \geq \delta$, so there exists R_2 with $\mu_1(R_2) \cdot |2\Pr_{\mu_1}(f=1 \mid R_2) - 1| \geq \delta$, implying the absolute mass removed is $\geq \delta \cdot (1-p_1)$. Continuing this way, the remaining mass after t steps is $\leq (1-\delta)^t$. To make it $\leq \frac{1}{3}$, take $t = O\left(\frac{1}{\delta}\log 3\right)$.

The deterministic protocol for μ hardcodes $R_1, \ldots, R_t$ and their majority labels $b_1, \ldots, b_t$. For each i, Alice sends 1 if $x \in A_i$, else 0. If 0, proceed to next. If 1, Bob sends 1 if $y \in B_i$, else 0. If both 1, output b_i. If not, proceed. If no match after t steps, output 0. The communication is at most $2t = O\left(\frac{1}{\delta}\right)$. The error under μ is at most the remaining mass $\leq \frac{1}{3}$. □

F Problems in Chapter 4

F.1 Problem 1

F.1.1 Problem Statement

Problem F.1 (Open Problem 4.1)**.** *Can $D(f, g)$ be smaller than $D(f) + D(g)$? How much smaller can it be? How much smaller can $D(f^\ell)$ be compared to $\ell \cdot D(f)$?*

F.1.2 GPT

Proof. We are tasked with investigating the relationship between the quantities $D(f, g)$, $D(f)$, $D(g)$, and $D(f^\ell)$, specifically addressing whether $D(f, g)$ can be smaller than $D(f) + D(g)$, and how much smaller $D(f^\ell)$ can be compared to $\ell \cdot D(f)$.

First, let us define the terms more precisely. Let $D(f)$ denote some measure of "distance" or "discrepancy" associated with the function f, and similarly for g. The quantity $D(f, g)$ represents the "distance" or "discrepancy" between the functions f and g. We aim to explore whether it is possible for $D(f, g)$ to be smaller than the sum $D(f) + D(g)$.

1. **Can $D(f, g)$ be smaller than $D(f) + D(g)$?**

We begin by considering a general property of metrics or distance measures. If D is a distance function, then it satisfies the triangle inequality, which states that for any functions f and g, we have:

$$D(f, g) \leq D(f) + D(g).$$

This inequality implies that $D(f, g)$ cannot exceed $D(f) + D(g)$. However, the question asks whether $D(f, g)$ can be strictly smaller than $D(f) + D(g)$.

In general, the triangle inequality is not an equality unless f and g are "closely related" in some sense, such as when $f = g$ or when f and g are very similar. Therefore, it is indeed possible for $D(f, g)$ to be smaller than $D(f) + D(g)$, and the difference between $D(f) + D(g)$ and $D(f, g)$ depends on the degree of similarity between f and g. If f and g are close in some sense (e.g., their values are very similar over their domain), then the discrepancy $D(f, g)$ may be significantly smaller than the sum $D(f) + D(g)$.

2. **How much smaller can $D(f, g)$ be compared to $D(f) + D(g)$?**

To explore how much smaller $D(f, g)$ can be, we note that the triangle inequality suggests that the smallest possible value for $D(f, g)$ occurs when f and g are identical, in which case:

$$D(f, g) = D(f) = D(g),$$

and thus

$$D(f, g) = D(f) + D(g) \quad \text{if and only if} \quad f = g.$$

Therefore, in the case of complete similarity between f and g, the difference between $D(f, g)$ and $D(f) + D(g)$ can be arbitrarily close to zero. More formally, for any $\epsilon > 0$, there exist functions f and g such that:

$$D(f, g) \leq D(f) + D(g) - \epsilon.$$

Thus, $D(f, g)$ can be arbitrarily smaller than $D(f) + D(g)$ when f and g are sufficiently similar.

3. **How much smaller can $D(f^\ell)$ be compared to $\ell \cdot D(f)$?**

Next, we examine the behavior of $D(f^\ell)$ in comparison to $\ell \cdot D(f)$ for some integer $\ell \geq 1$. By definition, $D(f^\ell)$ represents the distance associated with the function f^ℓ, which is typically a function of the form $f(x)^\ell$.

We apply the following property of distance measures: if the distance measure D is homogeneous of degree 1 (which is the case for many common distance functions, such as the L^p norms), then for any scalar α, we have:

$$D(f^\ell) = \ell \cdot D(f).$$

This homogeneity property means that, for such distance measures, $D(f^\ell)$ is exactly equal to $\ell \cdot D(f)$. Therefore, under the assumption of homogeneity, we conclude that $D(f^\ell)$ cannot be smaller than $\ell \cdot D(f)$, as they are equal.

If D is not homogeneous, then the relationship between $D(f^\ell)$ and $\ell \cdot D(f)$ depends on the specific nature of the distance function. However, for many standard distance functions, such as L^p norms, the equality holds, and we have:

$$D(f^\ell) = \ell \cdot D(f).$$

Thus, $D(f^\ell)$ is exactly $\ell \cdot D(f)$, and it cannot be smaller unless the distance function possesses some additional structure that alters this relationship.

□

F.1.3 Claude

Proof. We address the open problem by providing examples and establishing bounds for the certificate complexity of composed and iterated functions.

Part 1: Subadditivity of $D(f, g)$ versus $D(f) + D(g)$.

We first establish that $D(f, g) \leq D(f) + D(g)$ always holds. For any input (x, y) to $f \circ g$, a certificate can be constructed by combining a certificate for f at $g(x)$ and certificates for each coordinate of g at x. This gives the upper bound.

To show that $D(f, g)$ can be strictly smaller than $D(f)+D(g)$, consider the following construction. Let $f = \text{OR}_n$ and $g = \text{AND}_n$, where $n \geq 2$. We have $D(\text{OR}_n) = D(\text{AND}_n) = n$, so $D(f) + D(g) = 2n$.

For the composition $h(x_1, \ldots, x_n) = \text{OR}_n(\text{AND}_n(x_{1,1}, \ldots, x_{1,n}), \ldots, \text{AND}_n(x_{n,1}, \ldots, x_{n,n}))$, consider the input where all $x_{i,j} = 1$. To certify that $h = 1$, it suffices to show that one of the AND gates outputs 1. This requires fixing all n bits in one block, giving a certificate of size n. Thus $D(h) \leq n + (n - 1) \cdot 0 = n$ in favorable cases.

More precisely, by the composition theorem for certificate complexity, we have

$$D(f \circ g) \leq D(f) \cdot \max_i D(g_i)$$

where g_i are the coordinate functions. In many cases, this product can be significantly smaller than the sum $D(f) + D(g)$.

For a concrete example showing strict inequality, let $f(x_1, x_2) = x_1 \vee x_2$ and $g(y) = y \wedge y = y$ (the identity). Then $D(f) = 1$, $D(g) = 1$, so $D(f) + D(g) = 2$. However, $f \circ g = f$, so $D(f, g) = D(f) = 1 < 2$.

The maximum savings is $D(f) + D(g) - D(f, g) \geq D(f) + D(g) - D(f) \cdot D(g)$, which can be as large as $\Omega(n)$ when $D(f), D(g) = O(\sqrt{n})$.

Part 2: Sublinearity of $D(f^\ell)$ versus $\ell \cdot D(f)$.

For the iterated composition $f^\ell = f \circ f \circ \cdots \circ f$ (ℓ times), we always have $D(f^\ell) \leq \ell \cdot D(f)$ by repeatedly applying certificates.

To show strict inequality is possible, consider $f = \text{OR}_2$. We have $D(f) = 1$. For $f^2(x_1, x_2, x_3, x_4) = (x_1 \vee x_2) \vee (x_3 \vee x_4)$, which is OR_4, we have $D(f^2) = 1 < 2 = 2 \cdot D(f)$.

More generally, for $f = \text{OR}_n$, we have $D(f^\ell) = 1$ for all $\ell \geq 1$, while $\ell \cdot D(f) = \ell$. Thus the ratio $\frac{D(f^\ell)}{\ell \cdot D(f)} = \frac{1}{\ell}$ can be arbitrarily small.

For balanced functions where such extreme behavior does not occur, the composition theorem gives

$$D(f^\ell) \leq D(f)^\ell$$

which can still be significantly smaller than $\ell \cdot D(f)$ when $D(f) \geq 2$ and ℓ is large, since $D(f)^\ell \ll \ell \cdot D(f)$ for large ℓ when $D(f) < \ell$.

In summary, both $D(f, g)$ and $D(f^\ell)$ can be substantially smaller than their naive upper bounds, with savings ranging from constant factors to exponential improvements depending on the structure of the functions involved. □

F.1.4 Gemini

Proof. Let $D(f)$ denote the deterministic decision tree complexity of a Boolean function $f : \{0,1\}^n \to \{0,1\}$. We consider the direct sum problem of computing the pair $(f(x), g(y))$ given disjoint inputs $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$. We prove that $D(f,g) = D(f) + D(g)$. This implies that $D(f,g)$ cannot be smaller than the sum of the individual complexities, and by induction, $D(f^\ell) = \ell \cdot D(f)$.

First, we establish the upper bound $D(f,g) \leq D(f) + D(g)$. Let T_f and T_g be optimal deterministic decision trees for f and g with depths $D(f)$ and $D(g)$ respectively. We construct a tree T for the joint problem by replacing every leaf of T_f with a copy of T_g. The depth of T is exactly $D(f) + D(g)$, and it correctly computes both values. Thus, $D(f,g) \leq D(f) + D(g)$.

Next, we prove the lower bound $D(f,g) \geq D(f)+D(g)$. We rely on the following property of deterministic complexity: for any function h and any variable z, the inequality $D(h) \leq 1 + \max_{b\in\{0,1\}} D(h|_{z=b})$ holds. This is because an optimal tree for h can query z at the root and then proceed with optimal trees for the restrictions. Rearranging this, we have that for any variable z, there exists a value $b \in \{0,1\}$ such that

$$D(h|_{z=b}) \geq D(h) - 1.$$

Let T be an optimal decision tree computing $(f(x), g(y))$ with depth $d = D(f,g)$. We construct a path from the root to a leaf in T that maintains high residual complexity for both functions. Let v_0 be the root of T. We define a sequence of nodes $v_0, v_1, \ldots, v_k$ where v_k is a leaf, and a sequence of restrictions on the inputs. Let f_i and g_i denote the restrictions of f and g consistent with the path to v_i. Initially, $f_0 = f$ and $g_0 = g$.

At step i, suppose the node v_i queries a variable z. If z is a variable from the input x, then g is unaffected (so $g_{i+1} = g_i$). We choose the branch $b \in \{0,1\}$ such that $D(f_i|_{z=b}) \geq D(f_i) - 1$. We set v_{i+1} to be the child corresponding to b and $f_{i+1} = f_i|_{z=b}$. Consequently,

$$D(f_{i+1}) + D(g_{i+1}) \geq (D(f_i) - 1) + D(g_i) = D(f_i) + D(g_i) - 1.$$

Similarly, if z is a variable from the input y, then f is unaffected. We choose the branch b such that $D(g_i|_{z=b}) \geq D(g_i) - 1$. Setting $g_{i+1} = g_i|_{z=b}$ and $f_{i+1} = f_i$, we obtain

$$D(f_{i+1}) + D(g_{i+1}) \geq D(f_i) + (D(g_i) - 1) = D(f_i) + D(g_i) - 1.$$

We repeat this process until we reach a leaf v_k. At the leaf, the tree must determine the values of both f and g. This implies that f_k and g_k must be constant functions over the restricted subcube. The deterministic complexity of a constant function is 0. Therefore,

$$D(f_k) + D(g_k) = 0.$$

Summing the decrease in potential along the path of length k, we have:

$$D(f_k) + D(g_k) \geq D(f_0) + D(g_0) - k.$$

Substituting the boundary values:

$$0 \geq D(f) + D(g) - k \implies k \geq D(f) + D(g).$$

Since the depth of the tree is at least the length of this path ($d \geq k$), we conclude $D(f,g) \geq D(f) + D(g)$.

Combining the upper and lower bounds, we have $D(f,g) = D(f) + D(g)$. By induction on ℓ, it follows that $D(f^\ell) = \ell \cdot D(f)$. Thus, the complexity cannot be smaller than the sum of the individual complexities. □

F.1.5 Grok

Proof. The questions posed in Open Problem 4.1 concern the direct sum property in deterministic two-party communication complexity. For functions $f, g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ (typically considered on disjoint input domains when computing both simultaneously), $D(f, g)$ denotes the deterministic communication complexity of the combined function that outputs both $f(x_1, y_1)$ and $g(x_2, y_2)$. Similarly, $D(f^\ell)$ is the complexity of computing ℓ independent instances of f simultaneously.

It is always true that $D(f, g) \leq D(f) + D(g)$ and $D(f^\ell) \leq \ell \cdot D(f)$, achieved by executing optimal protocols for each instance sequentially and independently.

The questions ask whether strict inequality can hold and, if so, by how much.

This was a long-standing open problem, with the "direct sum conjecture" positing that no significant savings are possible, i.e., $D(f^\ell) \geq \ell \cdot (D(f) - o(1))$ or similar subadditivity bounds for all total functions.

However, the conjecture has been refuted. There exists a family of total functions $(f_n)_{n\in\mathbb{N}}$ such that for all natural numbers N, L, C, there exist $n \geq N$ and $\ell \geq L$ satisfying

$$D(f_n) \geq \frac{D(f_n^\ell)}{\ell} + C.$$

This implies

$$D(f_n^\ell) \leq \ell \cdot (D(f_n) - C),$$

so the total communication savings when computing ℓ instances jointly is at least $\ell \cdot C$, which can be made arbitrarily large by choice of parameters.

Thus, $D(f^\ell)$ can be smaller than $\ell \cdot D(f)$ by an arbitrarily large amount (both additively and by an arbitrary constant factor in the amortized per-copy cost). The same holds for $D(f, g)$ with g another copy of f (or suitably related functions), yielding savings over $D(f) + D(g)$.

Concrete constructions achieve constant-factor amortized savings even for fixed moderate ℓ (e.g., $\ell = 178$).

These counterexamples rely on carefully designed functions that force alternating communication rounds, exploiting the integrality of communication complexity to waste bits when protocols are executed independently but save them when interlaced jointly. □

F.2 Problem 2

F.2.1 Problem Statement

Problem F.2. *Can $D(f \wedge g)$ be smaller than $D(f) + D(g)$? How much smaller?*

F.2.2 GPT

Proof. Let f and g be two functions, and let $D(f)$ and $D(g)$ denote some measure of complexity or difficulty of f and g, respectively. The goal is to analyze whether $D(f \wedge g)$ can be smaller than $D(f) + D(g)$, and if so, how much smaller it can be.

We begin by recalling the definition of $D(f)$ and $D(g)$. These represent some non-negative quantities that measure the complexity of f and g, respectively. In general, $D(f \wedge g)$ is the measure of the complexity of the pointwise minimum of f and g. Specifically, for each x in the domain, we have the relation:

$$(f \wedge g)(x) = \min(f(x), g(x)).$$

Now, consider the following inequality for the complexity measure:

$$D(f \wedge g) \leq D(f) + D(g).$$

This inequality suggests that the complexity of the minimum function $f \wedge g$ is bounded by the sum of the complexities of f and g. To see if the inequality can be strict, we need to investigate if it is possible for $D(f \wedge g)$ to be smaller than $D(f) + D(g)$.

To understand this, we analyze a specific scenario. Suppose that f and g are functions where $f(x)$ and $g(x)$ are equal for most values of x, i.e., $f(x) = g(x)$ for a large portion of the domain. In this case, we have:

$$(f \wedge g)(x) = \min(f(x), g(x)) = f(x) = g(x).$$

Thus, $f \wedge g$ is identical to either f or g, and in this case, it follows that:

$$D(f \wedge g) = D(f) = D(g).$$

Therefore, in this situation, we have:

$$D(f \wedge g) = D(f) = D(g) \quad \text{and} \quad D(f) + D(g) = 2D(f).$$

Clearly, in this case, $D(f \wedge g)$ is strictly smaller than $D(f) + D(g)$, specifically by a factor of 2.

To examine the smallest possible difference, consider the scenario where f and g are completely disjoint, i.e., for all x in the domain, either $f(x) = 0$ and $g(x) = 1$, or $f(x) = 1$ and $g(x) = 0$. In this case, we have:

$$(f \wedge g)(x) = \min(f(x), g(x)) = 0 \quad \text{for all } x,$$

so $D(f \wedge g) = 0$, while $D(f) + D(g)$ is the sum of the complexities of f and g. Hence, in this case, we have:

$$D(f \wedge g) = 0 \quad \text{and} \quad D(f) + D(g) > 0.$$

Thus, $D(f \wedge g)$ can indeed be strictly smaller than $D(f) + D(g)$, and in fact, it can be arbitrarily smaller depending on the specific functions f and g.

Therefore, $D(f \wedge g)$ can be smaller than $D(f) + D(g)$, and the amount by which it is smaller depends on the specific properties of f and g. In the extreme case where f and g are disjoint, $D(f \wedge g)$ can be as small as 0, while $D(f) + D(g)$ remains positive.

□

F.2.3 Claude

Proof. We investigate whether $D(f \wedge g)$ can be smaller than $D(f) + D(g)$, where D denotes the deterministic decision tree complexity and $f \wedge g$ denotes the AND of Boolean functions f and g.

First, we establish that $D(f \wedge g) \leq D(f) + D(g)$. To compute $f \wedge g$, we can first compute f using at most $D(f)$ queries. If $f(x) = 0$, then $(f \wedge g)(x) = 0$ and we are done. If $f(x) = 1$, we then compute g using at most $D(g)$ additional queries, and $(f \wedge g)(x) = g(x)$. Thus $D(f \wedge g) \leq D(f) + D(g)$.

Now we show that $D(f \wedge g)$ can indeed be strictly smaller than $D(f) + D(g)$.

Construction of an example: Let $n = 2$ and define $f, g : \{0,1\}^2 \to \{0,1\}$ by $f(x_1, x_2) = x_1$ and $g(x_1, x_2) = x_2$. Then $D(f) = 1$ since we need only query x_1, and similarly $D(g) = 1$ since we need only query x_2. Therefore $D(f) + D(g) = 2$.

For $f \wedge g$, we have $(f \wedge g)(x_1, x_2) = x_1 \wedge x_2$. To compute this function, we must query both variables in the worst case. Specifically, if we query x_1 first and find $x_1 = 1$, we must still query x_2 to determine the output. Similarly, if we query x_2 first and find $x_2 = 1$, we must query x_1. Thus $D(f \wedge g) = 2$.

In this example, $D(f \wedge g) = D(f) + D(g)$, so there is no savings.

Better example: Consider $f, g : \{0,1\}^3 \to \{0,1\}$ defined by $f(x_1, x_2, x_3) = x_1 \vee x_2$ and $g(x_1, x_2, x_3) = x_1 \vee x_3$. We have $D(f) = 2$ (worst case: query x_1, if 0 then query x_2) and $D(g) = 2$, so $D(f) + D(g) = 4$.

Now $(f \wedge g)(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge (x_1 \vee x_3) = x_1 \vee (x_2 \wedge x_3)$ by distributivity. To compute this, we can query x_1 first. If $x_1 = 1$, output 1. If $x_1 = 0$, query both x_2 and x_3 to compute $x_2 \wedge x_3$. Thus $D(f \wedge g) = 3$.

Therefore $D(f \wedge g) = 3 < 4 = D(f) + D(g)$, showing a savings of 1.

Maximum savings: In general, we have the lower bound $D(f \wedge g) \geq \max\{D(f), D(g)\}$ since computing $f \wedge g$ requires determining at least one of f or g in certain cases. The maximum savings is therefore at most $D(f) + D(g) - \max\{D(f), D(g)\} = \min\{D(f), D(g)\}$.

This bound is tight: for $f = g$ with $D(f) = D(g) = k$, we have $D(f \wedge g) = D(f) = k$ while $D(f) + D(g) = 2k$, giving a savings of exactly $k = \min\{D(f), D(g)\}$.

Thus $D(f \wedge g)$ can be as much as $\min\{D(f), D(g)\}$ smaller than $D(f) + D(g)$. □

F.2.4 Gemini

Proof. Let $D(f)$ denote the deterministic decision tree complexity of a Boolean function $f : \{0,1\}^n \to \{0,1\}$. We investigate whether $D(f \wedge g) < D(f) + D(g)$ and the magnitude of the difference .

First, we observe that for any Boolean functions f and g, the inequality $D(f \wedge g) \leq D(f) + D(g)$ holds. A decision tree for $f \wedge g$ can be constructed by first evaluating f; if $f(x) = 0$, the value of $(f \wedge g)(x)$ is 0. If $f(x) = 1$, the tree proceeds to evaluate g. The depth of this combined tree is at most $D(f) + D(g)$.

To determine if $D(f \wedge g)$ can be strictly smaller than $D(f) + D(g)$, consider the case where f and g share variables. Let $n \geq 1$ and let $f : \{0,1\}^n \to \{0,1\}$ be the parity function, defined as $f(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$. The decision tree complexity of the parity function is $D(f) = n$, as the value of f depends on all n variables.

Let $g : \{0,1\}^n \to \{0,1\}$ be the negation of f, defined as $g(x) = \neg f(x)$. Similarly, $D(g) = n$. Consider the conjunction $h = f \wedge g$. For every input $x \in \{0,1\}^n$, we have $h(x) = f(x) \wedge \neg f(x) = 0$. Thus, h is the constant zero function, and its decision tree complexity is $D(f \wedge g) = 0$.

Comparing the quantities, we have $D(f) + D(g) = n + n = 2n$, while $D(f \wedge g) = 0$. The difference is $2n$. Since $D(f)$ and $D(g)$ are bounded by n, the sum $D(f) + D(g)$ is bounded by $2n$. Therefore, $D(f \wedge g)$ can be smaller than $D(f) + D(g)$ by the maximum possible amount, which is $D(f) + D(g)$.

We note that if f and g depend on disjoint sets of variables, the equality $D(f \wedge g) = D(f) + D(g)$ holds. However, in the general case where variables may be shared, the difference can be as large as the sum of the individual complexities. □

F.2.5 Grok

Proof. The question concerns whether the deterministic communication complexity of the conjunction $f \wedge g$, defined on disjoint inputs (Alice holds (x_f, x_g), Bob holds (y_f, y_g), and $(f \wedge g)((x_f, x_g), (y_f, y_g)) = f(x_f, y_f) \wedge g(x_g, y_g)$), can be strictly smaller than $D(f) + D(g)$, and if so, by how much.

The trivial upper bound is $D(f \wedge g) \leq D(f) + D(g)$, obtained by running optimal protocols for f and g sequentially (computing one fully, and the other only if necessary).

To show strict inequality is possible, consider the equality function EQ_n on n-bit strings, where $D(\mathrm{EQ}_n) = n + 1$. A protocol achieving this is: Alice sends her n-bit input (n bits), Bob sends a 1-bit comparison result (1 bit); both then know the output.

Now let $f = g = \mathrm{EQ}_n$ on disjoint inputs (total input size $2n$ bits each). Then $f \wedge g = \mathrm{EQ}_{2n}$, since the output is 1 if and only if both blocks match (i.e., the full $2n$-bit strings are equal). Thus $D(f \wedge g) = D(\mathrm{EQ}_{2n}) = 2n + 1$. Meanwhile, $D(f) + D(g) = (n + 1) + (n + 1) = 2n + 2$. Hence $D(f \wedge g) = 2n + 1 < 2n + 2 = D(f) + D(g)$, a strict savings of 1 bit.

More generally, for ℓ independent copies of EQ_n, the conjunction is $\mathrm{EQ}_{\ell n}$ with $D = \ell n + 1$, while the sum is $\ell(n + 1) = \ell n + \ell$, yielding savings of $\ell - 1$ bits additively (or amortized savings of nearly 1 bit per copy).

However, the amortized per-copy savings approaches 0 as $\ell \to \infty$, since $D(\bigwedge^{\ell} \mathrm{EQ}_n)/\ell = n + 1/\ell \approx n$, compared to $D(\mathrm{EQ}_n) = n + 1$.

Known lower bounds show that savings are limited in general. For suitable notions of complexity (e.g., on promises or specific classes), results establish $D(f \wedge g) \geq D(f) + D(g) - O(\log n)$ or similar polylogarithmic losses, implying that $D(f \wedge g)$ cannot be substantially smaller than $D(f) + D(g)$ for arbitrary f, g (savings at most polylogarithmic in the input size).

Thus, $D(f \wedge g)$ can be smaller than $D(f) + D(g)$, but the difference is bounded (at most $O(\log n)$ in known bounds, and at least $\Omega(1)$ in examples like equality). □

F.3 Problem 3

F.3.1 Problem Statement

Problem F.3 (Open Problem 4.6)**.** *Does there exist ℓ such that $R^{pub}(\ell^t) < \ell \cdot R^{pub}(t)$? How big can the gap be?*

F.3.2 GPT

Proof. We are tasked with determining whether there exists an ℓ such that $R^{\text{pub}}(\ell^t) < \ell \cdot R^{\text{pub}}(t)$ for a given function R^{pub}. Additionally, we need to assess the maximal possible gap between $R^{\text{pub}}(\ell^t)$ and $\ell \cdot R^{\text{pub}}(t)$.

Assume, for contradiction, that there exists an ℓ such that $R^{\text{pub}}(\ell^t) < \ell \cdot R^{\text{pub}}(t)$. This implies the existence of a value ℓ for which the following inequality holds:

$$R^{\text{pub}}(\ell^t) < \ell \cdot R^{\text{pub}}(t)$$

Let us now investigate the behavior of R^{pub} and the implications of this inequality. We first explore the properties of the function R^{pub}. By assumption, R^{pub} is a function that represents some measure of resource or time complexity, potentially in the context of cryptographic protocols or computational problems. Without loss of generality, assume that $R^{\text{pub}}(x)$ is a monotonically increasing function for positive values of x. This assumption is reasonable given the typical behavior of such functions in cryptographic contexts, where higher input values generally correspond to greater resource requirements.

We now examine the consequences of the inequality $R^{\text{pub}}(\ell^t) < \ell \cdot R^{\text{pub}}(t)$ under the assumption that R^{pub} is monotonically increasing. For any $\ell > 1$, we observe the following:

$$\ell^t > t \quad \text{and thus} \quad R^{\text{pub}}(\ell^t) > R^{\text{pub}}(t)$$

Since $\ell > 1$, this implies that $R^{\text{pub}}(\ell^t)$ grows faster than $R^{\text{pub}}(t)$ as t increases. Therefore, for the inequality $R^{\text{pub}}(\ell^t) < \ell \cdot R^{\text{pub}}(t)$ to hold, there must be a significant gap between $R^{\text{pub}}(\ell^t)$ and $\ell \cdot R^{\text{pub}}(t)$, which contradicts the expected growth of a monotonically increasing function.

Next, we consider the maximal possible gap between $R^{\text{pub}}(\ell^t)$ and $\ell \cdot R^{\text{pub}}(t)$. Suppose that the function R^{pub} grows faster than linearly with respect to its argument. In this case, the ratio of $R^{\text{pub}}(\ell^t)$ to $\ell \cdot R^{\text{pub}}(t)$ would increase as t grows, further reinforcing the impossibility of the inequality holding for large enough t. Thus, the gap between $R^{\text{pub}}(\ell^t)$ and $\ell \cdot R^{\text{pub}}(t)$ cannot grow arbitrarily large without violating the fundamental properties of R^{pub}.

Consequently, no such ℓ can exist that satisfies the inequality $R^{\text{pub}}(\ell^t) < \ell \cdot R^{\text{pub}}(t)$ for all t. The maximal gap between $R^{\text{pub}}(\ell^t)$ and $\ell \cdot R^{\text{pub}}(t)$ is constrained by the growth properties of R^{pub}, and the gap cannot exceed a certain bound.

Therefore, we conclude that such an ℓ does not exist, and the maximal gap between $R^{\text{pub}}(\ell^t)$ and $\ell \cdot R^{\text{pub}}(t)$ is finite. □

F.3.3 Claude

Proof. We provide a constructive answer to this open problem by demonstrating the existence of ℓ satisfying the desired inequality and establishing bounds on the gap.

Existence of such ℓ: We show that $\ell = 2$ satisfies $R^{\text{pub}}(2^t) < 2 \cdot R^{\text{pub}}(t)$ for sufficiently large t.

Consider the probabilistic method applied to public-coin protocols. For any function $f : \{0,1\}^{2^t} \to \{0,1\}$, we have by the direct sum property that

$$R^{\text{pub}}(f^{\otimes 2^t}) \leq 2^t \cdot R^{\text{pub}}(f) + O(\log(2^t)) = 2^t \cdot R^{\text{pub}}(f) + O(t).$$

Now consider the composition $g = f \circ f$ where $f : \{0,1\}^t \to \{0,1\}$ has $R^{\text{pub}}(f) = \Theta(t)$. By information complexity arguments, we have

$$R^{\text{pub}}(g) = R^{\text{pub}}(f \circ f) \leq R^{\text{pub}}(f) + R^{\text{pub}}(f) + o(t) = 2R^{\text{pub}}(f) + o(t).$$

However, the input size of g is $|g| = t^2$, not 2^t. To obtain the exponential gap, we use the following construction:

Let $h_t : \{0,1\}^{2^t} \to \{0,1\}$ be defined via a complete binary tree of depth t, where each internal node computes a function with public-coin complexity $\Theta(1)$. By the tree structure, we have

$$R^{\text{pub}}(h_t) = \Theta(t).$$

For the composed function $h_{2t} : \{0,1\}^{2^{2t}} \to \{0,1\}$, we similarly obtain

$$R^{\text{pub}}(h_{2t}) = \Theta(2t) = 2 \cdot \Theta(t) = 2 \cdot R^{\text{pub}}(h_t).$$

Since $2^{2t} = (2^t)^2 = 4^t$, setting $\ell = 4$ and $n = 2^t$, we have

$$R^{\text{pub}}(\ell^t) = R^{\text{pub}}(4^t) = R^{\text{pub}}(2^{2t}) = 2t < 4t - 2t = 4 \cdot t - 2t = \ell \cdot R^{\text{pub}}(2^t) - 2t.$$

For sufficiently large t, this gives $R^{\text{pub}}(4^t) < 4 \cdot R^{\text{pub}}(2^t)$, establishing existence.

Bounds on the gap: The multiplicative gap is defined as

$$\gamma(\ell, t) = \frac{\ell \cdot R^{\text{pub}}(t) - R^{\text{pub}}(\ell^t)}{R^{\text{pub}}(t)}.$$

From our construction, we have $\gamma(4, 2^t) \geq \frac{2t}{t} = 2$ for the tree-based protocol family.

More generally, using amortization arguments from communication complexity, for any $\ell \geq 2$ and appropriate function families, the gap can be as large as

$$\gamma(\ell, t) = \Omega(\log \ell).$$

This follows from the fact that $R^{\text{pub}}(\ell^t) = O(t \log \ell)$ for certain composed functions, while $\ell \cdot R^{\text{pub}}(t) = \Theta(\ell t)$, yielding a gap of $\Theta(\ell t) - O(t \log \ell) = \Theta(\ell t)$ when $\ell \gg \log \ell$.

Therefore, there exist values of ℓ (specifically $\ell \geq 2$) for which the strict inequality holds, and the gap can be as large as $\Omega(\ell \cdot R^{\text{pub}}(t))$ in absolute terms, or $\Omega(\ell)$ as a multiplicative factor. □

F.3.4 Gemini

Proof. We address the problem of the existence and magnitude of the gap between the randomized public-coin communication complexity of the direct sum of a task, $R^{\text{pub}}(\ell^t)$, and the linear scaling of the single-shot complexity, $\ell \cdot R^{\text{pub}}(t)$. We interpret ℓ^t as the task t composed ℓ times (the direct sum t^ℓ), where inputs are provided for ℓ independent instances and the goal is to compute the vector of results with bounded global error.

Answer: Yes, such an ℓ exists. The gap can be exponential in the complexity of t.

To prove this, we utilize the connection between randomized communication complexity and Information Complexity (IC). We consider the case where t is a partial function (a promise problem), for which strong separations have been established.

Let μ be a distribution over the inputs of t. The internal information complexity $IC_\mu(t, \varepsilon)$ is the minimum amount of information Alice and Bob must reveal to each other about their inputs to compute t with error ε under μ. We define $IC(t) = \sup_\mu IC_\mu(t, 1/3)$.

A fundamental property of information complexity is its additivity. For ℓ independent copies of t, we have:

$$IC(t^\ell) = \ell \cdot IC(t).$$

Furthermore, a celebrated result by Braverman (2012) relates the amortized communication complexity to information complexity. Specifically, for any task t and sufficiently large ℓ:

$$\lim_{\ell \to \infty} \frac{R^{\text{pub}}(t^\ell)}{\ell} = IC(t).$$

This implies that for large ℓ, $R^{\text{pub}}(t^\ell) \approx \ell \cdot IC(t)$.

To demonstrate a gap, we must identify a task t where the single-shot communication complexity $R^{\text{pub}}(t)$ is significantly larger than its information complexity $IC(t)$.

Ganor, Kol, and Raz (2016) proved the existence of a partial function t (specifically, a "bursting" variant of the Index function) exhibiting an exponential separation between communication complexity and information complexity. They showed that there exists a family of partial functions t_n such that:

$$IC(t_n) = O(n),$$

while

$$R^{\text{pub}}(t_n) \geq 2^{\Omega(n)}.$$

Let us fix such a function t. We compare $R^{\text{pub}}(t^\ell)$ with $\ell \cdot R^{\text{pub}}(t)$. Using the compression result for the direct sum, for sufficiently large ℓ:

$$R^{\text{pub}}(t^\ell) \leq \ell \cdot IC(t) \cdot (1 + o(1)).$$

Substituting the bounds for t:

$$R^{\text{pub}}(t^\ell) \leq \ell \cdot O(n).$$

However, the linear scaling of the single-shot complexity yields:

$$\ell \cdot R^{\text{pub}}(t) \geq \ell \cdot 2^{\Omega(n)}.$$

Thus, we have the strict inequality:

$$R^{\text{pub}}(t^\ell) \ll \ell \cdot R^{\text{pub}}(t).$$

The magnitude of the gap is exponential in the parameter n (which determines the complexity of t). Specifically, the ratio satisfies:

$$\frac{\ell \cdot R^{\text{pub}}(t)}{R^{\text{pub}}(t^\ell)} = \Omega\left(\frac{2^{\Omega(n)}}{n}\right).$$

Therefore, not only does such an ℓ exist, but the gap between the amortized cost and the single-shot cost can be exponentially large. □

F.3.5 Grok

Proof. The open problem asks whether there exists a positive integer ℓ such that the public-coin randomized communication complexity of the conjunction of ℓ independent copies of a given function t satisfies $R^{\text{pub}}(\bigwedge^\ell t) < \ell \cdot R^{\text{pub}}(t)$, and if so, how large the savings (gap) can be. Here, ℓ^t denotes the ℓ-fold conjunction of t on disjoint inputs (total input size ℓ times larger).

The trivial upper bound is $R^{\text{pub}}(\bigwedge^\ell t) \leq \ell \cdot R^{\text{pub}}(t)$, achieved by running ℓ independent optimal protocols sequentially (short-circuiting if an early copy evaluates to 0, though for worst-case total functions this yields at most constant-factor improvement for small ℓ).

However, strict inequality holds for $\ell \geq 2$, and the gap can be arbitrarily large.

Consider $t = \text{EQ}_m$, the equality function on m-bit strings, with $R^{\text{pub}}(\text{EQ}_m) = O(1)$ (using public randomness to select a universal hash function; Alice sends the hash of her input, Bob verifies). The ℓ-fold conjunction $\bigwedge^\ell \text{EQ}_m$ is equivalent to $\text{EQ}_{\ell m}$ on ℓm-bit strings, so $R^{\text{pub}}(\bigwedge^\ell \text{EQ}_m) = O(1)$. Thus, $O(1) < \ell \cdot O(1)$ for all $\ell \geq 2$, with additive gap $\Omega(\ell)$ (arbitrarily large as ℓ grows).

For a function with superconstant complexity, consider $t = \text{DISJ}_m$ (disjointness on sets of size m), with $R^{\text{pub}}(\text{DISJ}_m) = \Theta(\sqrt{m})$. The ℓ-fold conjunction $\bigwedge^\ell \text{DISJ}_m$ is equivalent to $\text{DISJ}_{\ell m}$ (the union has empty intersection iff every pair has empty intersection), so $R^{\text{pub}}(\bigwedge^\ell \text{DISJ}_m) = \Theta(\sqrt{\ell m}) = O(\sqrt{\ell} \cdot \sqrt{m})$. Meanwhile, $\ell \cdot R^{\text{pub}}(\text{DISJ}_m) = \Theta(\ell\sqrt{m})$. The ratio is $\Theta(\ell\sqrt{m}/(\sqrt{\ell}\sqrt{m})) = \Theta(\sqrt{\ell})$, so the total communication savings is a factor of $\sqrt{\ell}$ (arbitrarily large as ℓ increases).

In both examples, the gap grows unbounded with ℓ. Whether even larger (e.g., exponential) gaps are possible for suitable functions t, or whether there are inherent limitations beyond these polynomial-in-ℓ factors for certain classes, remains of interest, but the problem confirms existence of savings with arbitrarily large gaps. □

F.4 Problem 4

F.4.1 Problem Statement

Theorem F.4. *Show that $R^{pub}(\ell^t) = O(\ell \cdot \log \ell \cdot R^{pub}(t))$, for all t.*

F.4.2 GPT

Proof. We are tasked with showing that $R^{\text{pub}}(\ell^t) = O(\ell \cdot \log \ell \cdot R^{\text{pub}}(t))$ for all t.

To prove this, we begin by considering the definition of the function $R^{\text{pub}}(x)$. Let $R^{\text{pub}}(x)$ represent a function that grows asymptotically at most as some polynomial function of x. More specifically, we assume that there exists a constant $C > 0$ such that:

$$R^{\text{pub}}(x) \leq C \cdot x^k \quad \text{for some constant } k > 0.$$

Now, substituting $x = \ell^t$ into this inequality, we obtain:

$$R^{\text{pub}}(\ell^t) \leq C \cdot (\ell^t)^k = C \cdot \ell^{kt}.$$

Next, consider the right-hand side of the desired inequality, $\ell \cdot \log \ell \cdot R^{\text{pub}}(t)$. By the assumption on the growth rate of $R^{\text{pub}}(t)$, there exists a constant $D > 0$ such that:

$$R^{\text{pub}}(t) \leq D \cdot t^m \quad \text{for some constant } m > 0.$$

Thus, the right-hand side becomes:

$$\ell \cdot \log \ell \cdot R^{\text{pub}}(t) \leq \ell \cdot \log \ell \cdot D \cdot t^m.$$

To establish the desired inequality, we need to show that:

$$C \cdot \ell^{kt} \leq \ell \cdot \log \ell \cdot D \cdot t^m.$$

This inequality holds for sufficiently large values of ℓ and t, since for large ℓ, ℓ^{kt} grows asymptotically faster than $\ell \cdot \log \ell \cdot t^m$. Specifically, for large enough ℓ, we can choose constants such that the inequality holds, implying that:

$$R^{\text{pub}}(\ell^t) = O(\ell \cdot \log \ell \cdot R^{\text{pub}}(t)).$$

Therefore, the proof is complete. □

F.4.3 Claude

Proof. We prove that $R^{\text{pub}}(\ell^t) = O(\ell \cdot \log \ell \cdot R^{\text{pub}}(t))$ for all t.

Let $\mathcal{A}$ be an optimal public-coin protocol for computing ℓ^t with communication cost $R^{\text{pub}}(\ell^t)$. We construct a protocol for ℓ^t by composing protocols for t in a tree structure.

Consider the binary representation of ℓ. We compute ℓ^t by repeated squaring and multiplication. Specifically, write $\ell = \sum_{i=0}^{k} b_i 2^i$ where $b_i \in \{0, 1\}$ and $k = \lfloor \log_2 \ell \rfloor$. Then

$$\ell^t = \prod_{i=0}^{k} (2^i)^{t \cdot b_i} = \prod_{i=0}^{k} (t^{2^i})^{b_i}.$$

We compute t^{2^i} for $i = 0, 1, \ldots, k$ using a sequence of squaring operations:

$$\begin{aligned} t^{2^0} &= t, \\ t^{2^1} &= (t^{2^0})^2, \\ t^{2^2} &= (t^{2^1})^2, \\ &\vdots \\ t^{2^k} &= (t^{2^{k-1}})^2. \end{aligned}$$

Each squaring operation s^2 can be computed using a protocol for t by having Alice input (s, s) and Bob input (s, s) to compute the product. This requires communication $R^{\text{pub}}(t)$. Since we perform $k = \lfloor \log_2 \ell \rfloor$ squaring operations, the total cost for computing all t^{2^i} is at most $\log_2 \ell \cdot R^{\text{pub}}(t)$.

Next, we compute $\ell^t = \prod_{i:b_i=1} t^{2^i}$ by multiplying at most $k+1 \leq \log_2 \ell+1$ terms. Each multiplication of two values can be performed using a protocol for t with cost $R^{\text{pub}}(t)$. The number of multiplications needed is at most $\ell - 1$ (the number of ones in the binary representation of ℓ minus one), which is bounded by ℓ.

Alternatively, we can organize the multiplications as a balanced binary tree of depth $O(\log \ell)$, where each internal node represents a multiplication. This gives at most $\ell - 1$ multiplications, each costing $R^{\text{pub}}(t)$.

Combining both phases, the total communication cost is bounded by:

$$R^{\text{pub}}(\ell^t) \leq \log_2 \ell \cdot R^{\text{pub}}(t) + \ell \cdot R^{\text{pub}}(t) = O(\ell \cdot \log \ell \cdot R^{\text{pub}}(t)).$$

Since $\ell \geq 1$, we have $\log \ell \leq \ell$, and thus:

$$R^{\text{pub}}(\ell^t) = O(\ell \cdot \log \ell \cdot R^{\text{pub}}(t)).$$

□

F.4.4 Gemini

Proof. Let $t : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a function. We interpret the notation ℓ^t as the direct sum function consisting of ℓ independent copies of t. Specifically, let the input to ℓ^t be $(\vec{x}, \vec{y})$, where $\vec{x} = (x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$ and $\vec{y} = (y_1, \ldots, y_\ell) \in \mathcal{Y}^\ell$. The function is defined as $\ell^t(\vec{x}, \vec{y}) = (t(x_1, y_1), \ldots, t(x_\ell, y_\ell))$.

Let $R^{\text{pub}}(t)$ denote the randomized public-coin communication complexity of t with a maximum error probability of $1/3$. Let Π be an optimal public-coin protocol for t such that the communication cost is $C = R^{\text{pub}}(t)$ and the error probability is at most $1/3$ for every input.

To bound $R^{\text{pub}}(\ell^t)$, we construct a protocol Ψ that computes the vector of results with a global error probability of at most $1/3$. Since the output is a vector of length ℓ, the protocol succeeds only if every individual coordinate is computed correctly. To ensure the global error is bounded by $1/3$, we must reduce the error probability of the protocol for each individual instance.

Let $\delta = \frac{1}{3\ell}$. We construct an amplified protocol Π' for the single instance t by running the original protocol Π independently k times and outputting the majority result. By standard error reduction results using Chernoff bounds, the error probability decreases exponentially with the number of repetitions. Specifically, there exists a constant c_0 such that choosing $k = c_0 \log(1/\delta)$ ensures the error of Π' is at most δ. Substituting our choice of δ, we require $k = O(\log(3\ell)) = O(\log \ell)$ repetitions.

The communication cost of Π' is $k \cdot C = O(C \log \ell)$.

We define the protocol Ψ for the direct sum ℓ^t as follows: The players Alice and Bob run the protocol Π' independently for each coordinate $i \in \{1, \ldots, \ell\}$ on inputs (x_i, y_i). The total communication cost of Ψ is the sum of the costs of the ℓ invocations of Π':

$$\text{Cost}(\Psi) = \ell \cdot O(C \log \ell) = O(\ell \cdot \log \ell \cdot R^{\text{pub}}(t)).$$

We now analyze the success probability of Ψ. Let E_i be the event that the i-th coordinate $t(x_i, y_i)$ is computed incorrectly by Π'. By our construction of Π', $\Pr[E_i] \leq \delta = \frac{1}{3\ell}$. The protocol Ψ fails if there exists at least one index i such that the output is incorrect. Let E be the event that Ψ fails. By the union bound:

$$\Pr[E] = \Pr\left[\bigcup_{i=1}^{\ell} E_i\right] \leq \sum_{i=1}^{\ell} \Pr[E_i] \leq \sum_{i=1}^{\ell} \frac{1}{3\ell} = \ell \cdot \frac{1}{3\ell} = \frac{1}{3}.$$

Thus, Ψ computes ℓ^t with error probability at most $1/3$. Since Ψ is a valid randomized public-coin protocol, the complexity $R^{\text{pub}}(\ell^t)$ is bounded by the cost of Ψ.

Therefore, $R^{\text{pub}}(\ell^t) = O(\ell \cdot \log \ell \cdot R^{\text{pub}}(t))$. □

F.4.5 Grok

Proof. Let $c = R^{\text{pub}}(t)$. There exists a public-coin randomized protocol Π computing t with communication cost at most c such that for every input, the probability over the public randomness that Π outputs the correct value is at least $2/3$.

By standard error reduction via independent repetitions and majority vote, there exists a constant $K > 0$ and a public-coin protocol Π' computing t with communication cost at most $Kc \log \ell$ such that for every input, the error probability of Π' is at most $1/(100\ell)$. To achieve this, run $\Theta(\log \ell)$ independent executions of Π (using disjoint portions of the public random string) and output the majority vote; concentration bounds (Chernoff) ensure the desired error probability for sufficiently large constant in the $\Theta(\log \ell)$.

Now define a public-coin protocol for ℓ^t (the function requiring computation of ℓ disjoint instances of t and outputting 1 if and only if all instances evaluate to 1) as follows. The ℓ instances are ordered arbitrarily as $1, \ldots, \ell$.

For $i = 1$ to ℓ: - Execute Π' on the i-th instance. - If Π' outputs 0, output 0 (for the overall function) and halt. If all ℓ executions of Π' output 1, output 1.

The communication cost is at most $\ell \cdot Kc \log \ell = O(\ell \log \ell \cdot R^{\text{pub}}(t))$, as in the worst case all ℓ executions are performed.

For correctness, consider an arbitrary input to ℓ^t.

If the correct output is 1 (all instances of t evaluate to 1), then the protocol errs only if at least one execution of Π' outputs 0 (wrongly). By union bound, the error probability is at most $\ell \cdot (1/(100\ell)) = 1/100 < 1/3$.

If the correct output is 0 (at least one instance, say the j-th, evaluates to 0), then the protocol errs only if it outputs 1, which requires all executions of Π' (including the j-th) to output 1. In particular, this requires Π' on the j-th instance to output 1 wrongly, which occurs with probability at most $1/(100\ell) < 1/3$.

Thus, in both cases the error probability is less than $1/3$. Since the target bounded-error probability is usually $1/3$ (up to constants), and here it is strictly smaller, the protocol computes ℓ^t with bounded error. Therefore, $R^{\text{pub}}(\ell^t) = O(\ell \log \ell \cdot R^{\text{pub}}(t))$. □

F.5 Problem 5

F.5.1 Problem Statement

Problem F.5. *Express B_ε^1 as a linear program. Use the duality theorem of linear programming to write its* dual *program.*

Use the dual program to solve the next two parts of this exercise.

F.5.2 GPT

Proof. Let B_ε^1 denote the unit ball in the Euclidean space $\mathbb{R}^n$, i.e.,

$$B_\varepsilon^1 = \{x \in \mathbb{R}^n : \|x\|_1 \leq \varepsilon\},$$

where $\|x\|_1 = \sum_{i=1}^n |x_i|$ is the L_1 norm of the vector $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ and $\varepsilon > 0$ is a constant.

To express B_ε^1 as a linear program, we need to formulate the problem as a linear optimization problem. We aim to find a linear program whose feasible region is exactly B_ε^1. Consider the following linear programming formulation:

$$\text{Maximize} \quad 0 \quad \text{subject to} \quad \sum_{i=1}^n |x_i| \leq \varepsilon, \quad x \in \mathbb{R}^n.$$

Since the objective function is a constant (0), the goal of the linear program is simply to determine the feasible region of x, which is the set B_ε^1. To further work with this formulation, we recognize that the $|x_i|$ terms are non-linear. To linearize the program, we introduce auxiliary variables t_i such that $t_i \geq x_i$ and $t_i \geq -x_i$ for all $i = 1, 2, \ldots, n$. This gives the equivalent linear program:

$$\text{Maximize} \quad 0 \quad \text{subject to} \quad \sum_{i=1}^{n} t_i \leq \varepsilon, \quad t_i \geq x_i, \quad t_i \geq -x_i, \quad x \in \mathbb{R}^n, \quad t \in \mathbb{R}^n.$$

This is a linear program that expresses the unit ball B_ε^1.

Next, we formulate the dual of this linear program. The dual program is constructed by associating a dual variable $y_j \geq 0$ for each of the constraints. Specifically, for the constraint $\sum_{i=1}^{n} t_i \leq \varepsilon$, we introduce the dual variable $y_1 \geq 0$. The dual variables corresponding to the inequalities $t_i \geq x_i$ and $t_i \geq -x_i$ are denoted by $y_2 \geq 0$ and $y_3 \geq 0$, respectively.

The dual of the primal problem is given by:

$$\text{Minimize} \quad \varepsilon y_1 \quad \text{subject to} \quad y_1 - y_2 - y_3 = 0, \quad y_2 \geq 0, \quad y_3 \geq 0.$$

Thus, the dual linear program is:

$$\text{Minimize} \quad \varepsilon y_1 \quad \text{subject to} \quad y_1 - y_2 - y_3 = 0, \quad y_2 \geq 0, \quad y_3 \geq 0.$$

This dual formulation completes the expression of B_ε^1 as a linear program and the derivation of its dual program. □

F.5.3 Claude

Proof. We express B_ε^1 as a linear program and derive its dual using the duality theorem of linear programming.

Primal Program: The ℓ^1 ball $B_\varepsilon^1 = \{x \in \mathbb{R}^n : \|x\|_1 \leq \varepsilon\}$ can be expressed as the feasible region of a linear program. We introduce auxiliary variables $u_i, v_i \geq 0$ for $i = 1, \ldots, n$ such that $x_i = u_i - v_i$ and $|x_i| = u_i + v_i$. The primal linear program is:

$$\begin{aligned} \text{minimize} \quad & 0 \\ \text{subject to} \quad & \sum_{i=1}^{n} (u_i + v_i) \leq \varepsilon \\ & x_i = u_i - v_i, \quad i = 1, \ldots, n \\ & u_i, v_i \geq 0, \quad i = 1, \ldots, n \end{aligned}$$

Equivalently, we can write the primal program in standard form. For a linear objective $c^T x$ over B_ε^1:

$$\begin{aligned} \text{maximize} \quad & c^T x \\ \text{subject to} \quad & \sum_{i=1}^{n} |x_i| \leq \varepsilon \\ & x \in \mathbb{R}^n \end{aligned}$$

Using the decomposition $x_i = u_i - v_i$ with $u_i, v_i \geq 0$, this becomes:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} c_i (u_i - v_i) \\ \text{subject to} \quad & \sum_{i=1}^{n} (u_i + v_i) \leq \varepsilon \\ & u_i, v_i \geq 0, \quad i = 1, \ldots, n \end{aligned}$$

Dual Program: Applying the duality theorem of linear programming, we introduce a dual variable $\lambda \geq 0$ corresponding to the constraint $\sum_{i=1}^{n}(u_i + v_i) \leq \varepsilon$. The dual program is:

$$\begin{aligned} \text{minimize} \quad & \varepsilon\lambda \\ \text{subject to} \quad & \lambda \geq c_i, \quad i = 1, \ldots, n \\ & \lambda \geq -c_i, \quad i = 1, \ldots, n \\ & \lambda \geq 0 \end{aligned}$$

The constraints $\lambda \geq c_i$ and $\lambda \geq -c_i$ for all i are equivalent to $\lambda \geq |c_i|$ for all i, which is equivalent to $\lambda \geq \|c\|_\infty$. Therefore, the dual program simplifies to:

$$\begin{aligned} \text{minimize} \quad & \varepsilon\lambda \\ \text{subject to} \quad & \lambda \geq \|c\|_\infty \end{aligned}$$

The optimal solution is $\lambda^* = \|c\|_\infty$, giving the optimal value $\varepsilon\|c\|_\infty$.

By strong duality, we have:

$$\max_{x \in B_\varepsilon^1} c^T x = \varepsilon\|c\|_\infty$$

This establishes the dual program and demonstrates that the maximum of a linear functional over B_ε^1 is ε times the ℓ^∞ norm of the coefficient vector. □

F.5.4 Gemini

Proof. We interpret the notation B_ε^1 as the L_1-minimization problem with L_∞-bounded residual (often referred to as the Basis Pursuit Denoising problem with L_∞ noise constraint). The problem is defined as:

$$(B_\varepsilon^1) \quad \min_{x \in \mathbb{R}^n} \|x\|_1 \quad \text{subject to} \quad \|Ax - b\|_\infty \leq \varepsilon,$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $\varepsilon \geq 0$.

1. Formulation as a Linear Program

To express (B_ε^1) as a linear program, we decompose the variable x into its positive and negative parts. Let $x = u - v$ where $u, v \in \mathbb{R}^n$ and $u \geq 0, v \geq 0$. The L_1 norm is then given by $\|x\|_1 = \sum_{j=1}^{n}(u_j + v_j) = \mathbf{1}^T u + \mathbf{1}^T v$, where $\mathbf{1}$ denotes the vector of all ones.

The constraint $\|Ax - b\|_\infty \leq \varepsilon$ is equivalent to the pair of linear inequalities:

$$-\varepsilon\mathbf{1} \leq Ax - b \leq \varepsilon\mathbf{1}.$$

Substituting $x = u - v$, we obtain:

$$A(u - v) - b \leq \varepsilon\mathbf{1} \implies Au - Av \leq b + \varepsilon\mathbf{1},$$

and

$$-(A(u - v) - b) \leq \varepsilon\mathbf{1} \implies -Au + Av \leq -b + \varepsilon\mathbf{1}.$$

Thus, the primal linear program (Primal) is:

$$\begin{aligned} \min_{u,v} \quad & \mathbf{1}^T u + \mathbf{1}^T v \\ \text{s.t.} \quad & Au - Av \leq b + \varepsilon\mathbf{1} \\ & -Au + Av \leq -b + \varepsilon\mathbf{1} \\ & u \geq 0, \quad v \geq 0. \end{aligned}$$

2. Derivation of the Dual Program

We introduce dual variables $\lambda_1 \in \mathbb{R}^m$ and $\lambda_2 \in \mathbb{R}^m$ corresponding to the two inequality constraints, with $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$. The Lagrangian function $\mathcal{L}(u, v, \lambda_1, \lambda_2)$ is:

$$\begin{aligned}\mathcal{L} &= \mathbf{1}^T u + \mathbf{1}^T v + \lambda_1^T (Au - Av - b - \varepsilon \mathbf{1}) + \lambda_2^T (-Au + Av + b - \varepsilon \mathbf{1}) \\ &= (\mathbf{1} + A^T \lambda_1 - A^T \lambda_2)^T u + (\mathbf{1} - A^T \lambda_1 + A^T \lambda_2)^T v - (b + \varepsilon \mathbf{1})^T \lambda_1 + (b - \varepsilon \mathbf{1})^T \lambda_2.\end{aligned}$$

The dual function is $g(\lambda_1, \lambda_2) = \inf_{u \geq 0, v \geq 0} \mathcal{L}(u, v, \lambda_1, \lambda_2)$. For the infimum to be finite (and not $-\infty$), the coefficients of u and v must be non-negative:

$$\mathbf{1} + A^T(\lambda_1 - \lambda_2) \geq 0 \quad \text{and} \quad \mathbf{1} - A^T(\lambda_1 - \lambda_2) \geq 0.$$

Let $y = \lambda_2 - \lambda_1$. The constraints become:

$$\mathbf{1} - A^T y \geq 0 \quad \text{and} \quad \mathbf{1} + A^T y \geq 0 \implies -\mathbf{1} \leq A^T y \leq \mathbf{1} \implies \|A^T y\|_\infty \leq 1.$$

The objective function of the dual is the constant term in the Lagrangian:

$$\begin{aligned}\text{Dual Obj} &= -b^T \lambda_1 - \varepsilon \mathbf{1}^T \lambda_1 + b^T \lambda_2 - \varepsilon \mathbf{1}^T \lambda_2 \\ &= b^T(\lambda_2 - \lambda_1) - \varepsilon \mathbf{1}^T (\lambda_1 + \lambda_2) \\ &= b^T y - \varepsilon \|\lambda_1 + \lambda_2\|_1 \quad (\text{since } \lambda_1, \lambda_2 \geq 0).\end{aligned}$$

To maximize this objective for a fixed $y = \lambda_2 - \lambda_1$, we must minimize $\|\lambda_1 + \lambda_2\|_1$. Since $\lambda_1, \lambda_2 \geq 0$, we have $\lambda_1 + \lambda_2 = |\lambda_2 - \lambda_1| + 2\min(\lambda_1, \lambda_2)$. The maximum is achieved when $\min(\lambda_1, \lambda_2) = 0$ (complementarity), yielding $\lambda_1 + \lambda_2 = |y|$. Thus, the term becomes $-\varepsilon \|y\|_1$.

The dual program is therefore:

$$\max_{y \in \mathbb{R}^m} \quad b^T y - \varepsilon \|y\|_1 \quad \text{subject to} \quad \|A^T y\|_\infty \leq 1.$$

□

F.5.5 Grok

Proof. The ℓ_1-ball is defined as

$$B_\varepsilon^1 = \left\{ u \in \mathbb{R}^n \;\middle|\; \sum_{j=1}^n |u_j| \leq \varepsilon \right\}.$$

To formulate the problem of maximizing a linear function $c^T u$ over B_ε^1 as a linear program, introduce nonnegative variables $v_j, w_j \geq 0$ for $j = 1, \ldots, n$ such that $u_j = v_j - w_j$ and $|u_j| = v_j + w_j$.

The primal linear program is

$$\begin{aligned}\max_{v,w} \quad & \sum_{j=1}^n c_j (v_j - w_j) \\ \text{subject to} \quad & \sum_{j=1}^n (v_j + w_j) \leq \varepsilon, \\ & v_j \geq 0, \quad w_j \geq 0 \quad \forall\, j = 1, \ldots, n.\end{aligned}$$

Associating the dual variable $\lambda \geq 0$ with the single constraint yields the dual linear program

$$\begin{aligned}\min_{\lambda} \quad & \varepsilon \lambda \\ \text{subject to} \quad & \lambda \geq c_j \quad \forall\, j = 1, \ldots, n, \\ & \lambda \geq -c_j \quad \forall\, j = 1, \ldots, n,\end{aligned}$$

$$\lambda \geq 0,$$

which is equivalent to

$$\begin{aligned} \min_{\lambda} \quad & \varepsilon\lambda \\ \text{subject to} \quad & \lambda \geq |c_j| \quad \forall\, j = 1, \ldots, n, \\ & \lambda \geq 0. \end{aligned}$$

□

F.6 Problem 6

F.6.1 Problem Statement

Lemma F.6. $B_+^I(f \wedge g) = B_+^I(f)B_+^I(g)$

F.6.2 GPT

Proof. Let f and g be functions in the appropriate function space, and let B_+^I denote the operation applied to these functions. We aim to prove that:

$$B_+^I(f \wedge g) = B_+^I(f)B_+^I(g)$$

To begin, recall the definition of the operation B_+^I as applied to a function. Let us assume that B_+^I is a bilinear operator or a similar operation that satisfies the distributive property over the pointwise minimum, $f \wedge g$. Thus, we have:

$$B_+^I(f \wedge g) = B_+^I\left(\min(f(x), g(x))\right)$$

By the properties of B_+^I, it is assumed that the operation B_+^I respects the pointwise minimum, meaning:

$$B_+^I\left(\min(f(x), g(x))\right) = B_+^I(f)(x) \cdot B_+^I(g)(x)$$

This follows from the assumption that B_+^I acts pointwise and respects the product of f and g when applied to their minimum. Therefore, we conclude that:

$$B_+^I(f \wedge g) = B_+^I(f)B_+^I(g)$$

Thus, the lemma is proven. □

F.6.3 Claude

Proof. We prove that $B_+^I(f \wedge g) = B_+^I(f)B_+^I(g)$ by direct computation using the definition of the positive boundary operator.

By definition, the positive boundary operator B_+^I applied to a function $f \wedge g$ is given by

$$B_+^I(f \wedge g) = \sum_{i \in I} \frac{\partial(f \wedge g)}{\partial x_i} \cdot \mathbf{1}_{\{\frac{\partial(f \wedge g)}{\partial x_i} > 0\}}.$$

Applying the product rule for differentiation to $f \wedge g$, we obtain

$$\frac{\partial(f \wedge g)}{\partial x_i} = \frac{\partial f}{\partial x_i} \wedge g + f \wedge \frac{\partial g}{\partial x_i}.$$

Since $f \wedge g$ represents the wedge product (exterior product), and the positive boundary operator respects the multiplicative structure of the algebra, we have

$$B_+^I(f \wedge g) = \sum_{i \in I} \left(\frac{\partial f}{\partial x_i} \wedge g + f \wedge \frac{\partial g}{\partial x_i} \right) \cdot \mathbf{1}_{\{\frac{\partial(f \wedge g)}{\partial x_i} > 0\}}.$$

By the Leibniz rule for the boundary operator on the exterior algebra, this can be rewritten as

$$B_+^I(f \wedge g) = B_+^I(f) \wedge g + (-1)^{\deg(f)} f \wedge B_+^I(g).$$

However, in the context where B_+^I acts as a derivation on the algebra and the wedge product is interpreted as ordinary multiplication in the appropriate algebraic structure, we have

$$B_+^I(f \wedge g) = \sum_{i \in I} \frac{\partial f}{\partial x_i} \cdot g \cdot \mathbf{1}_{\{\frac{\partial f}{\partial x_i} > 0\}} + \sum_{i \in I} f \cdot \frac{\partial g}{\partial x_i} \cdot \mathbf{1}_{\{\frac{\partial g}{\partial x_i} > 0\}}.$$

This simplifies to

$$B_+^I(f \wedge g) = g \cdot B_+^I(f) + f \cdot B_+^I(g) = B_+^I(f) \cdot g + f \cdot B_+^I(g).$$

By the multiplicative property of the positive boundary operator and the fact that it acts as a derivation satisfying the product rule in this algebraic setting, we conclude

$$B_+^I(f \wedge g) = B_+^I(f) B_+^I(g).$$

This completes the proof. □

F.6.4 Gemini

Proof. Let I be a finite index set representing the variables of the Boolean functions. We assume that f and g are Boolean functions defined on $\{0,1\}^I$ with disjoint supports. Specifically, let $J, K \subseteq I$ be a partition of I (i.e., $J \cup K = I$ and $J \cap K = \emptyset$) such that f depends only on variables in J and g depends only on variables in K.

The operator B_+^I denotes the multilinear extension (Bernstein-Bézier form) of a function $h : \{0,1\}^I \to \mathbb{R}$, defined by:

$$B_+^I(h)(\mathbf{x}) = \sum_{\mathbf{v} \in \{0,1\}^I} h(\mathbf{v}) \beta_{\mathbf{v}}(\mathbf{x}),$$

where $\mathbf{x} = (x_i)_{i \in I} \in [0,1]^I$, and the basis functions are given by $\beta_{\mathbf{v}}(\mathbf{x}) = \prod_{i \in I} b_{v_i}(x_i)$, with $b_1(t) = t$ and $b_0(t) = 1 - t$.

Any vector $\mathbf{v} \in \{0,1\}^I$ can be uniquely decomposed as $\mathbf{v} = (\mathbf{u}, \mathbf{w})$, where $\mathbf{u} \in \{0,1\}^J$ and $\mathbf{w} \in \{0,1\}^K$. Due to the disjoint supports, $f(\mathbf{v}) = f(\mathbf{u})$ and $g(\mathbf{v}) = g(\mathbf{w})$. The conjunction $f \wedge g$ corresponds to the product in the Boolean domain, so $(f \wedge g)(\mathbf{v}) = f(\mathbf{u}) g(\mathbf{w})$.

The basis function $\beta_{\mathbf{v}}(\mathbf{x})$ factors over the disjoint sets J and K:

$$\beta_{\mathbf{v}}(\mathbf{x}) = \left(\prod_{j \in J} b_{u_j}(x_j) \right) \left(\prod_{k \in K} b_{w_k}(x_k) \right) = \beta_{\mathbf{u}}(\mathbf{x}_J) \beta_{\mathbf{w}}(\mathbf{x}_K).$$

Substituting these into the definition of $B_+^I(f \wedge g)$:

$$\begin{aligned} B_+^I(f \wedge g)(\mathbf{x}) &= \sum_{\mathbf{u} \in \{0,1\}^J} \sum_{\mathbf{w} \in \{0,1\}^K} f(\mathbf{u}) g(\mathbf{w}) \beta_{\mathbf{u}}(\mathbf{x}_J) \beta_{\mathbf{w}}(\mathbf{x}_K) \\ &= \left(\sum_{\mathbf{u} \in \{0,1\}^J} f(\mathbf{u}) \beta_{\mathbf{u}}(\mathbf{x}_J) \right) \left(\sum_{\mathbf{w} \in \{0,1\}^K} g(\mathbf{w}) \beta_{\mathbf{w}}(\mathbf{x}_K) \right). \end{aligned}$$

Now consider $B_+^I(f)$. Using the partition of unity property $\sum_{\mathbf{w} \in \{0,1\}^K} \beta_{\mathbf{w}}(\mathbf{x}_K) = 1$, we have:

$$\begin{aligned} B_+^I(f)(\mathbf{x}) &= \sum_{\mathbf{u} \in \{0,1\}^J} \sum_{\mathbf{w} \in \{0,1\}^K} f(\mathbf{u}) \beta_{\mathbf{u}}(\mathbf{x}_J) \beta_{\mathbf{w}}(\mathbf{x}_K) \\ &= \left(\sum_{\mathbf{u} \in \{0,1\}^J} f(\mathbf{u}) \beta_{\mathbf{u}}(\mathbf{x}_J) \right) \left(\sum_{\mathbf{w} \in \{0,1\}^K} \beta_{\mathbf{w}}(\mathbf{x}_K) \right) \\ &= \sum_{\mathbf{u} \in \{0,1\}^J} f(\mathbf{u}) \beta_{\mathbf{u}}(\mathbf{x}_J). \end{aligned}$$

Similarly, $B_+^I(g)(\mathbf{x}) = \sum_{\mathbf{w}\in\{0,1\}^K} g(\mathbf{w})\beta_{\mathbf{w}}(\mathbf{x}_K)$. Comparing the factored form of $B_+^I(f \wedge g)$ with the expressions for $B_+^I(f)$ and $B_+^I(g)$, we obtain:

$$B_+^I(f \wedge g) = B_+^I(f)B_+^I(g).$$

□

F.6.5 Grok

Proof. Assume that f and g represent two independent systems with associated lifetimes T_f and T_g, respectively, and that $B_+^I(h)$ denotes the survival function of the system corresponding to h, i.e., $B_+^I(h)(t) = P(T_h > t)$ for $t \geq 0$.

The operation $f \wedge g$ corresponds to connecting the two systems in series, so the lifetime of the combined system is $T_{f\wedge g} = \min(T_f, T_g)$.

Then, for any $t \geq 0$,

$$B_+^I(f \wedge g)(t) = P(T_{f\wedge g} > t) = P(\min(T_f, T_g) > t) = P(T_f > t \cap T_g > t).$$

Since the systems are independent, the events $\{T_f > t\}$ and $\{T_g > t\}$ are independent, so

$$P(T_f > t \cap T_g > t) = P(T_f > t) \cdot P(T_g > t) = B_+^I(f)(t) \cdot B_+^I(g)(t).$$

This holds for all $t \geq 0$. Therefore, the survival functions satisfy

$$B_+^I(f \wedge g) = B_+^I(f) \cdot B_+^I(g),$$

where the product on the right is the pointwise product of functions. □

F.7 Problem 7

F.7.1 Problem Statement

Theorem F.7. *Let f be a Boolean function and let $R^{1,pub}(f)$ denote the complexity of computing f using a public coin protocol that makes a one-sided error. Then*

$$\log B_1^{\downarrow}(f) \leq R^{1,pub}(f) + O(1).$$

F.7.2 GPT

Proof. Let f be a Boolean function, and let $R^{1,\text{pub}}(f)$ denote the complexity of computing f using a public coin protocol that makes a one-sided error. We are tasked with proving the inequality

$$\log B_1^{\downarrow}(f) \leq R^{1,\text{pub}}(f) + O(1),$$

where $B_1^{\downarrow}(f)$ is the minimal number of bits required to compute f in the public coin model with one-sided error, and $R^{1,\text{pub}}(f)$ is the complexity in this model.

Step 1: Relating $B_1^{\downarrow}(f)$ and $R^{1,\text{pub}}(f)$ By definition, $B_1^{\downarrow}(f)$ represents the minimum number of bits that need to be communicated in a public coin protocol with one-sided error for computing f. On the other hand, $R^{1,\text{pub}}(f)$ denotes the complexity of computing f in a public coin protocol that allows one-sided error. These two measures are related because the complexity of computing f in a public coin protocol with one-sided error determines the number of bits exchanged during the protocol. Therefore, we expect that

$$B_1^{\downarrow}(f) \leq 2^{R^{1,\text{pub}}(f)}.$$

This is because each bit of communication can be associated with an individual computation step in the public coin protocol, and the protocol can make an error on one side.

Step 2: Taking the logarithm Taking the logarithm of both sides of the inequality, we obtain

$$\log B_1^{\downarrow}(f) \leq \log\left(2^{R^{1,\text{pub}}(f)}\right).$$

Using the property of logarithms $\log(a^b) = b\log(a)$, this simplifies to

$$\log B_1^{\downarrow}(f) \leq R^{1,\text{pub}}(f)\log 2.$$

Step 3: Accounting for the constant factor Since $\log 2$ is a constant, we can write

$$\log B_1^{\downarrow}(f) \leq R^{1,\text{pub}}(f) + O(1),$$

where the $O(1)$ term accounts for the constant factor introduced by $\log 2$.

Thus, we have shown that

$$\log B_1^{\downarrow}(f) \leq R^{1,\text{pub}}(f) + O(1),$$

as required.

□

F.7.3 Claude

Proof. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let π be a public coin protocol computing f with one-sided error and communication cost $R^{1,\text{pub}}(f)$.

By definition of public coin protocols, there exists a shared random string r of length $O(R^{1,\text{pub}}(f))$ such that for each fixed r, the protocol becomes deterministic. Let $\mathcal{R}$ denote the set of all possible random strings.

For each random string $r \in \mathcal{R}$, the deterministic protocol induced by r partitions the input space into at most $2^{R^{1,\text{pub}}(f)}$ rectangles, corresponding to the different transcripts of length at most $R^{1,\text{pub}}(f)$.

Since the protocol has one-sided error, without loss of generality assume that:

(i) If $f(x,y) = 1$, then $\Pr_r[\pi(x,y,r) = 1] \geq 2/3$

(ii) If $f(x,y) = 0$, then $\Pr_r[\pi(x,y,r) = 0] = 1$

For each (x,y) with $f(x,y) = 1$, there exists a set $\mathcal{R}_{x,y} \subseteq \mathcal{R}$ with $|\mathcal{R}_{x,y}| \geq (2/3)|\mathcal{R}|$ such that $\pi(x,y,r) = 1$ for all $r \in \mathcal{R}_{x,y}$.

By the one-sided error property, for each $r \in \mathcal{R}$, the rectangle R_r corresponding to any transcript that outputs 1 must be a 1-monochromatic rectangle (i.e., $f(x,y) = 1$ for all $(x,y) \in R_r$).

Consider the collection of all 1-monochromatic rectangles that arise from protocols with random strings of length at most $R^{1,\text{pub}}(f) + O(1)$. There are at most $2^{R^{1,\text{pub}}(f)+O(1)}$ such random strings, and for each random string, there are at most $2^{R^{1,\text{pub}}(f)}$ possible transcripts.

Therefore, the total number of 1-monochromatic rectangles needed is at most $2^{R^{1,\text{pub}}(f)+O(1)} \cdot 2^{R^{1,\text{pub}}(f)} = 2^{2R^{1,\text{pub}}(f)+O(1)}$.

However, by a more careful analysis using the probabilistic method, for each 1-input (x,y), at least a constant fraction of random strings r produce a transcript that covers (x,y) with a 1-monochromatic rectangle. By a covering argument, we can select a subcollection of at most $2^{R^{1,\text{pub}}(f)+O(1)}$ rectangles that cover all 1-inputs of f.

By definition, $B_1^{\downarrow}(f)$ is the minimum number of 1-monochromatic rectangles needed to cover all 1-inputs of f. Therefore:

$$B_1^{\downarrow}(f) \leq 2^{R^{1,\text{pub}}(f)+O(1)}.$$

Taking logarithms of both sides yields:

$$\log B_1^{\downarrow}(f) \leq R^{1,\text{pub}}(f) + O(1).$$

□

F.7.4 Gemini

Proof. Let $f : X \times Y \to \{0,1\}$ be a Boolean function. Let $R^{1,\text{pub}}(f)$ denote the public-coin randomized communication complexity of f with one-sided error. Let $B_1^{\downarrow}(f)$ denote the fractional covering number of 1-inputs of f by 1-monochromatic rectangles, which is the optimal value of the following linear program:

$$\begin{aligned} \text{minimize} \quad & \sum_{R \in \mathcal{R}^1_{mono}} w_R \\ \text{subject to} \quad & \sum_{R \ni (x,y)} w_R \geq 1 \quad \forall (x,y) \in f^{-1}(1) \\ & w_R \geq 0 \end{aligned}$$

where $\mathcal{R}^1_{mono}$ is the set of 1-monochromatic rectangles.

Let $C = R^{1,\text{pub}}(f)$. Then, there exists a public-coin communication protocol Π for f with cost C. Π defines a distribution μ over deterministic protocols $\{\pi_r\}_{r \in \Omega}$, where Ω is the public random coin space, such that for every 1-input (x,y), the probability that Π accepts (x,y) is at least $1/2$, and for every 0-input, the protocol rejects. Since π_r is a deterministic protocol of cost C, its accepting region $A_r = \pi_r^{-1}(1)$ can be written as a union of at most 2^C disjoint 1-monochromatic rectangles, denoted by $\mathcal{S}_r = \{R_{r,1}, \ldots, R_{r,k_r}\}$, where $k_r \leq 2^C$.

We construct a fractional cover as follows. For each rectangle $R \in \mathcal{R}^1_{mono}$, define its weight as $w_R = 2 \int_\Omega \mathbb{1}_{R \in \mathcal{S}_r} d\mu(r)$.

Consider any 1-input $(x,y) \in f^{-1}(1)$. Then,

$$\sum_{R \ni (x,y)} w_R = \sum_{R \ni (x,y)} 2 \int_\Omega \mathbb{1}_{R \in \mathcal{S}_r} d\mu(r) = 2 \int_\Omega \sum_{R \ni (x,y)} \mathbb{1}_{R \in \mathcal{S}_r} d\mu(r).$$

For a fixed r, $\sum_{R \ni (x,y)} \mathbb{1}_{R \in \mathcal{S}_r} = \mathbb{1}_{(x,y) \in A_r}$. Therefore, the above is equal to $2 \Pr[\Pi \text{ accepts } (x,y)] \geq 2 \cdot (1/2) = 1$. Thus, the constraint $\sum_{R \ni (x,y)} w_R \geq 1$ is satisfied.

The total weight of the covering is given by

$$\sum_R w_R = \sum_R 2 \int_\Omega \mathbb{1}_{R \in \mathcal{S}_r} d\mu(r) = 2 \int_\Omega \sum_R \mathbb{1}_{R \in \mathcal{S}_r} d\mu(r).$$

The inner sum is the number of accepting rectangles in π_r, which is $|\mathcal{S}_r| \leq 2^C$. Thus, the total weight is at most $2 \cdot 2^C \int d\mu(r) = 2^{C+1}$. Therefore, $B_1^{\downarrow}(f) \leq 2^{C+1}$, which implies $\log B_1^{\downarrow}(f) \leq R^{1,\text{pub}}(f) + 1$. □

Proving Communication Complexity Bound

I've now written the formal proof for this theorem. It centers on constructing a fractional rectangle cover from the public coin communication protocol. Key is linking the protocol's guarantees to a feasible solution, then bounding the weight, which directly proves the theorem.

Constructing a Formal Proof

I am now generating the precise LaTeX code. The plan is to define terms clearly, then construct a fractional rectangle cover. From there, I'll bound the cover's weight to show the desired communication complexity result. This will be a standard proof format.

</think>

Proof. Let $f : X \times Y \to \{0,1\}$ be a Boolean function. We define $B_1^{\downarrow}(f)$ as the one-sided fractional rectangle cover number of f. Formally, let $\mathcal{R}^1_{\text{mono}}$ be the set of all combinatorial rectangles $R \subseteq X \times Y$ such that $f(x,y) = 1$ for all $(x,y) \in R$. Then $B_1^{\downarrow}(f)$ is the optimal value of the following linear program:

$$\begin{aligned} \text{minimize} \quad & \sum_{R \in \mathcal{R}^1_{\text{mono}}} w_R \\ \text{subject to} \quad & \sum_{R \ni (x,y)} w_R \geq 1 \quad \forall (x,y) \in f^{-1}(1), \end{aligned}$$

$$w_R \geq 0 \quad \forall R \in \mathcal{R}^1_{\text{mono}}.$$

Let $C = R^{1,\text{pub}}(f)$. By definition, there exists a public-coin randomized protocol Π with communication cost C that computes f with one-sided error. Specifically, Π can be viewed as a probability distribution μ over a set of deterministic protocols $\{\pi_r\}_{r\in\Omega}$. The one-sided error property implies:

1. For all (x, y) such that $f(x, y) = 0$, $\Pr_{r\sim\mu}[\pi_r(x, y) = 1] = 0$.
2. For all (x, y) such that $f(x, y) = 1$, $\Pr_{r\sim\mu}[\pi_r(x, y) = 1] \geq 1/2$.

Condition 1 implies that for every $r \in \Omega$, if π_r accepts an input (x, y), then $f(x, y) = 1$. Thus, the set of inputs accepted by any deterministic protocol π_r in the support of μ is a subset of $f^{-1}(1)$.

For a fixed deterministic protocol π_r, let $L(\pi_r)$ denote the set of leaves of the protocol tree where the protocol outputs 1 (accepts). Each leaf $\ell \in L(\pi_r)$ corresponds to a rectangle $R_\ell \in \mathcal{R}^1_{\text{mono}}$. Since the depth of π_r is at most C, the number of such leaves is bounded by $|L(\pi_r)| \leq 2^C$. The set of inputs accepted by π_r is the disjoint union $\bigcup_{\ell\in L(\pi_r)} R_\ell$.

We now construct a feasible solution to the linear program for $B^{\downarrow}_1(f)$. For each $R \in \mathcal{R}^1_{\text{mono}}$, define the weight w_R as:

$$w_R = 2 \cdot \mathbb{E}_{r\sim\mu}\left[\sum_{\ell\in L(\pi_r)} \mathbb{I}(R_\ell = R)\right],$$

where $\mathbb{I}(\cdot)$ is the indicator function. Essentially, w_R is twice the expected number of times rectangle R appears as an accepting leaf in the randomized protocol.

First, we verify that these weights satisfy the covering constraint. Let (x, y) be an input such that $f(x, y) = 1$. We compute the total weight of rectangles covering (x, y):

$$\sum_{R\ni(x,y)} w_R = \sum_{R\ni(x,y)} 2 \cdot \mathbb{E}_{r\sim\mu}\left[\sum_{\ell\in L(\pi_r)} \mathbb{I}(R_\ell = R)\right].$$

By linearity of expectation, we can move the summation inside:

$$\sum_{R\ni(x,y)} w_R = 2 \cdot \mathbb{E}_{r\sim\mu}\left[\sum_{R\ni(x,y)} \sum_{\ell\in L(\pi_r)} \mathbb{I}(R_\ell = R)\right].$$

For a fixed r, the inner sum counts the number of accepting rectangles in π_r that contain (x, y). Since the rectangles associated with the leaves of a deterministic protocol are disjoint, (x, y) is contained in exactly one rectangle R_ℓ if π_r accepts (x, y), and zero otherwise. Thus:

$$\sum_{R\ni(x,y)} \sum_{\ell\in L(\pi_r)} \mathbb{I}(R_\ell = R) = \mathbb{I}(\pi_r \text{ accepts } (x, y)).$$

Substituting this back into the expectation:

$$\sum_{R\ni(x,y)} w_R = 2 \cdot \mathbb{E}_{r\sim\mu}[\mathbb{I}(\pi_r \text{ accepts } (x, y))] = 2 \cdot \Pr_{r\sim\mu}[\pi_r(x, y) = 1].$$

By the correctness condition of the protocol for 1-inputs, this probability is at least $1/2$. Therefore:

$$\sum_{R\ni(x,y)} w_R \geq 2 \cdot \frac{1}{2} = 1.$$

The constraints of the linear program are satisfied.

Next, we bound the objective value of this solution:

$$\sum_{R \in \mathcal{R}^1_{\text{mono}}} w_R = \sum_R 2 \cdot \mathbb{E}_{r \sim \mu} \left[\sum_{\ell \in L(\pi_r)} \mathbb{I}(R_\ell = R) \right].$$

Using linearity of expectation again:

$$\sum_R w_R = 2 \cdot \mathbb{E}_{r \sim \mu} \left[\sum_{\ell \in L(\pi_r)} \sum_R \mathbb{I}(R_\ell = R) \right] = 2 \cdot \mathbb{E}_{r \sim \mu}[|L(\pi_r)|].$$

Since every deterministic protocol π_r has depth at most C, the number of accepting leaves $|L(\pi_r)|$ is at most 2^C. Thus:

$$\sum_R w_R \leq 2 \cdot 2^C = 2^{C+1}.$$

Since $B_1^{\downarrow}(f)$ is the minimum possible value of the objective function, we have:

$$B_1^{\downarrow}(f) \leq 2^{C+1}.$$

Taking the logarithm of both sides:

$$\log B_1^{\downarrow}(f) \leq C + 1 = R^{1,\text{pub}}(f) + 1.$$

This confirms that $\log B_1^{\downarrow}(f) \leq R^{1,\text{pub}}(f) + O(1)$. □

F.7.5 Grok

Proof. Let $r = R^{1,\text{pub}}(f)$. There exists a public-coin randomized protocol Π computing f with one-sided error and communication cost at most r, such that Π always rejects on inputs with $f(x, y) = 0$ and accepts with probability at least $2/3$ on inputs with $f(x, y) = 1$.

The possible transcripts of Π are bit strings of total length at most r. There are fewer than 2^{r+1} such transcripts.

For each possible transcript t, define

$$R_t = \{(x, y) \mid \Pr[\Pi \text{ on } (x, y) \text{ produces transcript } t \text{ and accepts}] > 0\}.$$

Each $R_t \subseteq f^{-1}(1)$, since if $(x, y) \in R_t$ with $f(x, y) = 0$, then Π would accept with positive probability on an input where $f = 0$, contradicting the one-sided error property. Thus, each R_t is 1-monochromatic.

Furthermore, if $f(x, y) = 1$, then with probability at least $2/3 > 0$ the protocol accepts, so there exists some transcript t produced with positive probability upon acceptance. Hence $(x, y) \in R_t$ for some t.

The sets R_t therefore cover $f^{-1}(1)$. The number of such (possibly empty) sets is less than 2^{r+1}, so

$$B_1^{\downarrow}(f) \leq 2^{r+1} \leq 2^{R^{1,\text{pub}}(f)+1}.$$

Taking logarithms (base 2) yields

$$\log B_1^{\downarrow}(f) \leq R^{1,\text{pub}}(f) + 1 \leq R^{1,\text{pub}}(f) + O(1).$$

□

F.8 Problem 8

F.8.1 Problem Statement

Theorem F.8. *For all Boolean functions f,*

$$D(f^\ell) = \Omega(\ell(\sqrt{D(f)} - \log n - O(1))).$$

F.8.2 GPT

Proof. Let f be a Boolean function defined on n variables. The notation $D(f)$ represents the deterministic decision tree complexity of f, and f^ℓ is the function f composed with itself ℓ times. We are tasked with proving that:

$$D(f^\ell) = \Omega(\ell(\sqrt{D(f)} - \log n - O(1))).$$

Step 1: Basic Definitions Recall that the decision tree complexity $D(f)$ is the minimum number of queries to variables required to evaluate the Boolean function f. Let $D(f^\ell)$ denote the decision tree complexity of the function f^ℓ, which is the ℓ-fold composition of f.

Step 2: Generalization of Composition Complexity We start by analyzing the decision tree complexity of f^ℓ, which involves repeated application of the function f. Each application of f in the composition f^ℓ may depend on the values of previous computations. As such, we need to bound the complexity of evaluating f^ℓ in terms of $D(f)$.

Step 3: Relating $D(f^\ell)$ to $D(f)$ The decision tree complexity of the composition f^ℓ can be bounded by a recursive relation. Specifically, each query in the decision tree for f^ℓ can be viewed as a query in the decision tree for f applied to previous results, leading to the following recursion:

$$D(f^\ell) \leq \ell D(f).$$

This is a straightforward upper bound, as evaluating each iteration of the function requires at most $D(f)$ queries.

Step 4: Lower Bound on $D(f^\ell)$ To establish the lower bound, we turn to the well-known result from decision tree complexity theory that relates $D(f)$ to the depth of the tree and its combinatorial structure. Specifically, it has been shown that:

$$D(f) \geq \Omega(\sqrt{D(f)}).$$

This gives us a lower bound on the decision tree complexity of f itself. By applying this lower bound to the function composition f^ℓ, we get:

$$D(f^\ell) = \Omega(\ell(\sqrt{D(f)} - \log n - O(1))).$$

Step 5: Conclusion Thus, we have derived the desired lower bound for the decision tree complexity of f^ℓ:

$$D(f^\ell) = \Omega(\ell(\sqrt{D(f)} - \log n - O(1))).$$

This completes the proof. □

F.8.3 Claude

Proof. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let f^ℓ denote the ℓ-fold composition of f, where $f^\ell(x_1, \ldots, x_\ell) = f(f(\cdots f(x_1, x_2) \cdots, x_{\ell-1}), x_\ell)$ with appropriate input groupings.

We begin by applying the composition theorem for deterministic query complexity. For any Boolean function $g : \{0,1\}^m \to \{0,1\}$, we have the fundamental relation

$$D(g \circ f) \geq D(g) \cdot \text{bs}(f),$$

where $\text{bs}(f)$ denotes the block sensitivity of f.

By the well-known relationship between deterministic complexity and block sensitivity, we have

$$\text{bs}(f) = \Omega(\sqrt{D(f)}).$$

For the ℓ-fold composition, we apply induction. The base case $\ell = 1$ is trivial. For the inductive step, assume the result holds for $\ell - 1$. Then

$$D(f^\ell) = D(f^{\ell-1} \circ f) \geq D(f^{\ell-1}) \cdot \text{bs}(f).$$

By the inductive hypothesis and the block sensitivity lower bound,

$$D(f^\ell) \geq \Omega((\ell-1)(\sqrt{D(f)} - \log n - O(1))) \cdot \Omega(\sqrt{D(f)}).$$

However, we must be more careful with the composition. Using the refined composition theorem that accounts for the structure of iterated composition, we have

$$D(f^\ell) \geq \sum_{i=1}^{\ell} \text{bs}(f) = \ell \cdot \text{bs}(f).$$

Now, applying the certificate complexity lower bound, we know that for any function f,

$$\text{bs}(f) \geq \sqrt{D(f)} - \log n - O(1).$$

This follows from the relationship between block sensitivity, certificate complexity, and deterministic complexity, combined with the fact that $D(f) \leq C(f)^2$ where $C(f)$ is the certificate complexity, and $\text{bs}(f) \geq C(f)/(1 + \log n)$.

Combining these results, we obtain

$$D(f^\ell) \geq \ell \cdot \text{bs}(f) \geq \ell \cdot (\sqrt{D(f)} - \log n - O(1)).$$

Therefore,

$$D(f^\ell) = \Omega(\ell(\sqrt{D(f)} - \log n - O(1))),$$

as required. □

F.8.4 Gemini

Proof. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. We interpret f^ℓ as the direct sum of ℓ copies of f, defined on $\ell \cdot n$ variables. Specifically, let the input be $x = (x^{(1)}, \ldots, x^{(\ell)})$ where each $x^{(i)} \in \{0,1\}^n$. The function f^ℓ computes the vector of values $(f(x^{(1)}), \ldots, f(x^{(\ell)}))$. Alternatively, if f^ℓ represents the composition with a specific outer function $g : \{0,1\}^\ell \to \{0,1\}$ (such as XOR or AND) that depends on all ℓ variables, the deterministic query complexity satisfies the multiplicative property.

A fundamental result in deterministic query complexity states that for the direct sum (or the XOR composition), the complexity is additive. That is,

$$D(f^\ell) = \ell \cdot D(f).$$

To prove the theorem, it suffices to show that

$$\ell \cdot D(f) = \Omega\left(\ell\left(\sqrt{D(f)} - \log n - O(1)\right)\right).$$

Dividing by ℓ, we must show that

$$D(f) = \Omega\left(\sqrt{D(f)} - \log n - O(1)\right).$$

Let $d = D(f)$. Since f is a Boolean function on n variables, $0 \leq d \leq n$. We analyze the inequality $d \geq c(\sqrt{d} - \log n - k)$ for some constants $c, k > 0$.

Case 1: $d \leq 4\log^2 n$. In this case, for sufficiently large n, $\sqrt{d} \leq 2\log n$. Consequently, the term $\sqrt{d} - \log n - O(1)$ is bounded by $\log n - O(1)$. Since the lower bound in the Ω notation allows for an arbitrary multiplicative constant and the subtraction of a constant, and since the expression $\sqrt{d} - \log n$ may be negative or small, the condition is trivially satisfied or vacuous if the RHS is negative. Specifically, if $\sqrt{D(f)} < \log n + O(1)$, the RHS is negative, and since $D(f^\ell) \geq 0$, the inequality holds.

Case 2: $d > 4\log^2 n$. In this case, $\sqrt{d} > 2\log n$, which implies $\log n < \frac{1}{2}\sqrt{d}$. Then,

$$\sqrt{d} - \log n - O(1) < \sqrt{d} - \frac{1}{2}\sqrt{d} = \frac{1}{2}\sqrt{d}.$$

Since $d \geq 1$, we have $d \geq \sqrt{d}$. Therefore,

$$D(f) = d \geq \sqrt{d} > 2\left(\sqrt{d} - \log n - O(1)\right).$$

Thus, $D(f) = \Omega(\sqrt{D(f)} - \log n - O(1))$.

Multiplying by ℓ, we obtain

$$\ell D(f) = \Omega\left(\ell\left(\sqrt{D(f)} - \log n - O(1)\right)\right).$$

Substituting $D(f^\ell) = \ell D(f)$, we conclude

$$D(f^\ell) = \Omega\left(\ell\left(\sqrt{D(f)} - \log n - O(1)\right)\right).$$

□

F.8.5 Grok

Proof. Let $m = D(f)$ and assume for contradiction that $D(f^\ell) = d < \ell(\sqrt{m} - \log n - c)$ for a sufficiently large constant $c > 0$.

Set $s = \sqrt{m} - \log n - c$.

Then $d < \ell s$.

Consider an arbitrary root-to-leaf path in the decision tree, which has length at most d.

The path makes at most $d < \ell s$ queries in total across the ℓ copies.

By the pigeonhole principle, there exists at least one copy i such that the number of queries to copy i along this path is less than s.

The queries to copy i along this path, together with their answers, define a decision tree of depth less than $s < \sqrt{m} \leq m = D(f)$ for computing f on copy i (with the query choices possibly depending on answers in other copies, but the depth on copy i is less than s).

Since the depth is less than $D(f)$, this decision tree does not compute f correctly on all inputs to copy i.

Thus, there exists an assignment to the bits of copy i for which the output at the leaf for component i is incorrect for f on that assignment.

Set the bits of copy i to this assignment and set the bits of the other copies so that the answers to queries match those on the path.

The path is followed, the leaf is reached, and the output for component i is incorrect.

This contradicts the assumption that the decision tree computes f^ℓ correctly.

Therefore, $D(f^\ell) \geq \ell(\sqrt{D(f)} - \log n - O(1))$. □

F.9 Problem 9

F.9.1 Problem Statement

Theorem F.9. *Let f and g be matrices. Then* $\operatorname{rank}(f \wedge g) = \operatorname{rank}(f)\operatorname{rank}(g)$.

F.9.2 GPT

Proof. Let f and g be two matrices. We are tasked with proving that $\operatorname{rank}(f \wedge g) = \operatorname{rank}(f)\operatorname{rank}(g)$.

First, we recall the definition of the rank of a matrix. The rank of a matrix is the dimension of its row space, which is the same as the dimension of its column space. Therefore, for any matrix A, we have $\operatorname{rank}(A) = \dim(\operatorname{Row}(A)) = \dim(\operatorname{Col}(A))$.

Next, we examine the matrix $f \wedge g$. The symbol $\wedge$ represents the exterior product of f and g, which can be interpreted as the matrix formed by taking the wedge product of the rows or columns of f and g. More formally, we denote this operation as $f \wedge g = f \otimes g$, where $\otimes$ represents the Kronecker product.

We now explore the relationship between the ranks of the matrices f, g, and $f \wedge g$.

By the properties of the Kronecker product, we know that the rank of the Kronecker product satisfies the following identity:

$$\operatorname{rank}(f \otimes g) = \operatorname{rank}(f) \cdot \operatorname{rank}(g).$$

Since $f \wedge g$ is equivalent to $f \otimes g$ in this context, we can apply this result to conclude that:

$$\operatorname{rank}(f \wedge g) = \operatorname{rank}(f) \cdot \operatorname{rank}(g).$$

Thus, the rank of the matrix $f \wedge g$ is indeed equal to the product of the ranks of f and g, as required. □

F.9.3 Claude

Proof. Let $f : V \to W$ and $g : X \to Y$ be linear maps between vector spaces, which we represent as matrices. We interpret $f \wedge g$ as the tensor product (or Kronecker product) of the matrices f and g, denoted $f \otimes g$.

First, we establish that the tensor product of linear maps corresponds to the tensor product of vector spaces. Specifically, $f \otimes g : V \otimes X \to W \otimes Y$ is defined by $(f \otimes g)(v \otimes x) = f(v) \otimes g(x)$ for all $v \in V$ and $x \in X$, extended linearly.

Let $\{v_1, \ldots, v_r\}$ be a basis for $\operatorname{im}(f)$ where $r = \operatorname{rank}(f)$, and let $\{w_1, \ldots, w_s\}$ be a basis for $\operatorname{im}(g)$ where $s = \operatorname{rank}(g)$.

We claim that $\{v_i \otimes w_j : 1 \leq i \leq r, 1 \leq j \leq s\}$ forms a basis for $\operatorname{im}(f \otimes g)$.

To show this set spans $\operatorname{im}(f \otimes g)$, consider any element in $\operatorname{im}(f \otimes g)$. Such an element has the form $(f \otimes g)(\sum_k a_k \otimes b_k)$ for some $a_k \in V$ and $b_k \in X$. This equals $\sum_k f(a_k) \otimes g(b_k)$. Since $f(a_k) \in \operatorname{im}(f)$ and $g(b_k) \in \operatorname{im}(g)$, we can write $f(a_k) = \sum_i \alpha_{ki} v_i$ and $g(b_k) = \sum_j \beta_{kj} w_j$. Therefore,

$$\sum_k f(a_k) \otimes g(b_k) = \sum_k \sum_{i,j} \alpha_{ki} \beta_{kj} (v_i \otimes w_j),$$

which shows that $\{v_i \otimes w_j\}$ spans $\operatorname{im}(f \otimes g)$.

To show linear independence, suppose $\sum_{i,j} c_{ij}(v_i \otimes w_j) = 0$ for some scalars c_{ij}. This can be rewritten as $\sum_i v_i \otimes (\sum_j c_{ij} w_j) = 0$. Since $\{v_1, \ldots, v_r\}$ is linearly independent in W and $\{w_1, \ldots, w_s\}$ is linearly independent in Y, the set $\{v_i \otimes w_j\}$ is linearly independent in $W \otimes Y$ by the fundamental property of tensor products.

Therefore, $\{v_i \otimes w_j : 1 \leq i \leq r, 1 \leq j \leq s\}$ is a basis for $\operatorname{im}(f \otimes g)$, and this basis has exactly rs elements.

Thus, $\operatorname{rank}(f \otimes g) = rs = \operatorname{rank}(f) \cdot \operatorname{rank}(g)$. □

F.9.4 Gemini

Proof. Let f be an $m \times n$ matrix and g be a $p \times q$ matrix with entries in a field $\mathbb{F}$. We interpret the operation $f \wedge g$ as the Kronecker product of matrices. Let $r = \operatorname{rank}(f)$ and $s = \operatorname{rank}(g)$.

By the rank normal form theorem, there exist invertible matrices $P_1 \in GL_m(\mathbb{F})$, $Q_1 \in GL_n(\mathbb{F})$, $P_2 \in GL_p(\mathbb{F})$, and $Q_2 \in GL_q(\mathbb{F})$ such that

$$P_1 f Q_1 = \begin{pmatrix} I_r & 0 \\ 0 & 0 \end{pmatrix}_{m \times n} \quad \text{and} \quad P_2 g Q_2 = \begin{pmatrix} I_s & 0 \\ 0 & 0 \end{pmatrix}_{p \times q},$$

where I_k denotes the $k \times k$ identity matrix. Let $J_f = P_1 f Q_1$ and $J_g = P_2 g Q_2$.

Using the mixed-product property of the Kronecker product, which states that $(A \wedge B)(C \wedge D) = (AC) \wedge (BD)$ for matrices of compatible dimensions, we observe:

$$(P_1 \wedge P_2)(f \wedge g)(Q_1 \wedge Q_2) = (P_1 f Q_1) \wedge (P_2 g Q_2) = J_f \wedge J_g.$$

Since the Kronecker product of invertible matrices is invertible, the matrices $P_1 \wedge P_2$ and $Q_1 \wedge Q_2$ are invertible. Multiplication by invertible matrices preserves rank, therefore:

$$\operatorname{rank}(f \wedge g) = \operatorname{rank}(J_f \wedge J_g).$$

The matrix $J_f \wedge J_g$ is the Kronecker product of two diagonal matrices containing only 1s and 0s on their diagonals. Specifically, J_f has r ones and J_g has s ones. The resulting matrix $J_f \wedge J_g$ is a diagonal matrix

of size $mp \times nq$. A diagonal entry of $J_f \wedge J_g$ is non-zero (specifically equal to 1) if and only if it is the product of a non-zero diagonal entry of J_f and a non-zero diagonal entry of J_g.

Consequently, the number of non-zero entries on the diagonal of $J_f \wedge J_g$ is exactly the product of the number of non-zero entries in J_f and J_g. Thus, the rank of $J_f \wedge J_g$ is $r \cdot s$.

We conclude:

$$\operatorname{rank}(f \wedge g) = rs = \operatorname{rank}(f) \operatorname{rank}(g).$$

□

F.9.5 Grok

Proof. Let f be an $m \times n$ matrix over a field K, representing a linear map $f : K^n \to K^m$. Let g be a $p \times q$ matrix over K, representing a linear map $g : K^q \to K^p$. The matrix $f \wedge g$ is the Kronecker product, which is the matrix of the tensor product map $f \otimes g : K^n \otimes_K K^q \to K^m \otimes_K K^p$ with respect to the standard bases.

The rank of $f \wedge g$ is the dimension of the image of $f \otimes g$. We claim that $\operatorname{im}(f \otimes g) = \operatorname{im} f \otimes_K \operatorname{im} g$.

First, observe that $\operatorname{im}(f \otimes g) \subseteq \operatorname{im} f \otimes_K \operatorname{im} g$, since for any $x \in K^n$ and $y \in K^q$, $(f \otimes g)(x \otimes y) = f(x) \otimes g(y) \in \operatorname{im} f \otimes_K \operatorname{im} g$, and the image is spanned by such elements.

Conversely, let $u \in \operatorname{im} f$ and $v \in \operatorname{im} g$. Then there exist $a \in K^n$ and $b \in K^q$ such that $u = f(a)$ and $v = g(b)$. Thus, $u \otimes v = f(a) \otimes g(b) = (f \otimes g)(a \otimes b) \in \operatorname{im}(f \otimes g)$. Since the tensor product $\operatorname{im} f \otimes_K \operatorname{im} g$ is spanned by elements of the form $u \otimes v$ with $u \in \operatorname{im} f$ and $v \in \operatorname{im} g$, it follows that $\operatorname{im} f \otimes_K \operatorname{im} g \subseteq \operatorname{im}(f \otimes g)$.

Therefore, $\operatorname{im}(f \otimes g) = \operatorname{im} f \otimes_K \operatorname{im} g$. Taking dimensions over K, we obtain

$$\operatorname{rank}(f \wedge g) = \dim_K(\operatorname{im} f \otimes_K \operatorname{im} g) = \dim_K(\operatorname{im} f) \cdot \dim_K(\operatorname{im} g) = \operatorname{rank}(f) \cdot \operatorname{rank}(g),$$

as required. □

F.10 Problem 10

F.10.1 Problem Statement

Theorem F.10 (Exercise 4.14)**.** *$D(\varepsilon\alpha^\ell) \geq \ell \cdot n$, where this can be proved using a fooling set argument and also using Exercise 4.12.*

F.10.2 GPT

Proof. Let $\alpha = \mathrm{EQ}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be the equality function defined by $\mathrm{EQ}_n(x,y) = 1$ iff $x = y$, and let $\varepsilon = \mathrm{AND}_\ell : \{0,1\}^\ell \to \{0,1\}$ be the ℓ-ary AND, defined by $\mathrm{AND}_\ell(b_1, \ldots, b_\ell) = 1$ iff $b_i = 1$ for all $i \in [\ell]$. Define $f = \varepsilon\alpha^\ell$ on inputs $x = (x^{(1)}, \ldots, x^{(\ell)}) \in (\{0,1\}^n)^\ell$ and $y = (y^{(1)}, \ldots, y^{(\ell)}) \in (\{0,1\}^n)^\ell$ by

$$f(x,y) = \varepsilon\big(\alpha(x^{(1)}, y^{(1)}), \ldots, \alpha(x^{(\ell)}, y^{(\ell)})\big) = \bigwedge_{i=1}^{\ell} \mathrm{EQ}_n(x^{(i)}, y^{(i)}).$$

We prove $D(f) \geq \ell n$ by exhibiting a fooling set of size $2^{\ell n}$ for f. Let $m = \ell n$ and identify $(\{0,1\}^n)^\ell$ with $\{0,1\}^m$ via concatenation. Define

$$F = \{(z,z) : z \in \{0,1\}^m\} \subseteq \{0,1\}^m \times \{0,1\}^m.$$

For each $(z,z) \in F$, we have $x = y = z$, hence $x^{(i)} = y^{(i)}$ for all $i \in [\ell]$, so $\mathrm{EQ}_n(x^{(i)}, y^{(i)}) = 1$ for all i and thus $f(z,z) = 1$.

Now take two distinct elements $(z,z), (w,w) \in F$ with $z \neq w$. Then there exists an index $i \in [\ell]$ such that the ith n-bit block differs, i.e., $z^{(i)} \neq w^{(i)}$. Consider the mixed input (z,w). For this i, $\mathrm{EQ}_n(z^{(i)}, w^{(i)}) = 0$, hence

$$f(z,w) = \bigwedge_{j=1}^{\ell} \mathrm{EQ}_n(z^{(j)}, w^{(j)}) = 0.$$

Similarly, $f(w, z) = 0$. Therefore, for any two distinct pairs $(z, z), (w, w) \in F$, at least one of the cross pairs (z, w) or (w, z) (in fact both) is mapped to 0 by f, while $f(z, z) = f(w, w) = 1$. This shows that F is a 1-fooling set for f.

It is a standard fact that if a Boolean function f has a fooling set of size M, then its deterministic communication complexity satisfies $D(f) \geq \log_2 M$, since each monochromatic 1-rectangle can contain at most one element of a 1-fooling set and a deterministic protocol partitions the input space into at most $2^{D(f)}$ monochromatic rectangles. Applying this to F yields

$$D(f) \geq \log_2 |F| = \log_2(2^m) = m = \ell n.$$

Hence $D(\varepsilon\alpha^\ell) \geq \ell \cdot n$, as required. □

F.10.3 Claude

Proof. We prove that $D(\varepsilon\alpha^\ell) \geq \ell \cdot n$ using a fooling set argument.

Let $L = \varepsilon\alpha^\ell$ denote the language consisting of the empty string ε and the string α^ℓ, where α is a symbol from an alphabet of size n and $\ell \geq 1$.

We construct a fooling set S for L as follows. For each $i \in \{1, 2, \ldots, \ell\}$ and each symbol $a \in \Sigma$ where $|\Sigma| = n$, define the pair $(x_{i,a}, y_{i,a})$ where:

$$x_{i,a} = \alpha^{i-1} \quad \text{and} \quad y_{i,a} = a\alpha^{\ell-i}.$$

Note that $x_{i,a}y_{i,a} = \alpha^{i-1}a\alpha^{\ell-i}$. This string is in L if and only if $a = \alpha$ and $i - 1 + 1 + \ell - i = \ell$, which holds. Thus $x_{i,\alpha}y_{i,\alpha} = \alpha^\ell \in L$.

For distinct pairs $(i, a) \neq (j, b)$, we have $x_{i,a}y_{j,b} = \alpha^{i-1}b\alpha^{\ell-j}$. If $i \neq j$, then the length is $i - 1 + 1 + \ell - j = \ell + i - j \neq \ell$ (since $i \neq j$), so $x_{i,a}y_{j,b} \notin L$. If $i = j$ but $a \neq b$, then $x_{i,a}y_{i,b} = \alpha^{i-1}b\alpha^{\ell-i}$ has length ℓ but contains the symbol $b \neq \alpha$ at position i, so $x_{i,a}y_{i,b} \notin L$.

Therefore, $S = \{(x_{i,a}, y_{i,a}) : 1 \leq i \leq \ell, a \in \Sigma\}$ is a fooling set of size $\ell \cdot n$.

By the fooling set lower bound theorem, we have $D(L) \geq |S| = \ell \cdot n$.

Alternatively, using Exercise 4.12, which states that $D(L_1 \cup L_2) \geq D(L_1) + D(L_2)$ for disjoint languages L_1 and L_2, we can write $L = \{\varepsilon\} \cup \{\alpha^\ell\}$. Since $D(\{\varepsilon\}) = 1$ and by applying the union bound iteratively to decompose $\{\alpha^\ell\}$ into a union of languages that require distinguishing among n choices at each of ℓ positions, we obtain $D(\{\alpha^\ell\}) \geq \ell \cdot n - 1$. Combined with the singleton language $\{\varepsilon\}$, this yields $D(L) \geq \ell \cdot n$. □

F.10.4 Gemini

<think>**Examining the Theorem's Structure**

I'm currently breaking down the theorem statement, focusing on the core components like $D(\varepsilon\alpha^\ell)$, and how each part contributes to the overall inequality. The aim is to understand the implications of each symbol and their relationships, a necessary first step towards constructing a valid proof.

Investigating the Function's Behavior

I'm now diving deeper into the meaning of $\varepsilon\alpha^\ell$. Based on the "Tribes" function context and the suggestions from Kushilevitz and Nisan, I'm leaning toward α as a function related to "Tribes" or AND/OR composition. Considering the context of Exercises 4.12 and 4.14 in *Communication Complexity*, I am determining whether to utilize a "fooling set argument" or a similar technique to prove the given inequality.

Formulating Proof Strategy

I'm now contemplating specific proof techniques. Given the structure of the inequality and the communication complexity context, I'm leaning heavily towards an argument based on the direct sum theorem applied to the equality function. The goal is to leverage the communication complexity of equality, possibly using a reduction or, more likely, composition of the AND function and equality function to prove the given inequality regarding $D(\varepsilon\alpha^\ell)$. I am now focused on formally constructing a rigorous argument using the known properties of EQ_n.

Confirming the Compositional Structure

I'm now increasingly convinced that $\varepsilon\alpha^\ell$ is a composition of functions. I suspect α represents a fundamental gadget, perhaps AND or OR. The fact that the lower bound scales as $\ell \cdot n$ strongly suggests that the function's input is of size $O(\ell n)$, likely involving ℓ repeated instances of a base function on n bits. I'm moving forward with a proof strategy based on decomposing the given function.

Defining the Core Function

I'm now zeroing in on defining the precise function represented by $\varepsilon\alpha^\ell$. Based on the form of the inequality and the desired lower bound of $\ell \cdot n$, the function appears to operate on ℓ instances of a function with n input bits each. Given the context of the "Tribes" problem, the function is most likely the AND composition of ℓ Equality functions.

Defining the Core Function

I've determined that $\varepsilon\alpha^\ell$ most likely represents an AND composition of ℓ Equality functions, each operating on inputs of size n. This aligns with the expected lower bound of $\ell \cdot n$, where n represents the bit size for each equality comparison, and ℓ indicates the number of parallel comparisons. Now I need to formalize a proof for this structure.

Finalizing Function Decomposition

I'm now very confident that I've correctly interpreted the function structure. $\varepsilon\alpha^\ell$ is most likely ℓ Equality functions on n-bit inputs, combined with the AND operation. My next task is to carefully construct the formal proof to show this.

Finalizing the Formal Proof

I am now completely confident in the function decomposition, which is ℓ instances of Equality functions on n-bit inputs, with the AND operation. Now, I am prepared to write the formal proof, ensuring all definitions and arguments are meticulously defined.

Refining Proof Construction

I'm now putting together the complete LaTeX proof. I am writing out the function definitions, constructing the fooling set, and carefully showing the set's properties and size. I'm taking special care with notation to precisely represent the function composition. I am confident I'm on the right track.

Constructing Formal Definitions

I'm establishing a complete and rigorous proof by precisely defining all functions and mathematical objects. This involves formalizing the structure of $\varepsilon\alpha^\ell$ as the AND of ℓ Equality functions, each acting on n-bit inputs, to align with the prompt.

Refining the Proof's Arguments

I'm now carefully constructing the proof, making sure I address the context of communication complexity. My aim is to define the key functions precisely, construct a valid fooling set, and properly calculate its size. I'm focusing on rigorously deriving the lower bound, consistent with the prompt.

Constructing the Proof Structure

I'm now creating the essential definitions and laying out the proof steps. It will use a classic fooling set argument, a common tactic in communication complexity. This approach requires constructing and validating the fooling set. I'm taking care to align the notation with the prompt's $\varepsilon\alpha^\ell$ notation.

Constructing the LaTeX Proof

I am now generating the entire LaTeX proof. I am defining the function, constructing the appropriate fooling set, calculating the size, and then proving the lower bound on the communication complexity to ensure all conditions from the prompt are met.

Synthesizing the Formal Proof

I'm now fully prepared to write the LaTeX proof, meticulously defining the function, constructing the fooling set, calculating its size, and rigorously deriving the lower bound, all while adhering to the structure outlined in the prompt.

</think>

Proof. We interpret the notation $\varepsilon\alpha^\ell$ as the function $f : (\{0,1\}^n)^\ell \times (\{0,1\}^n)^\ell \to \{0,1\}$ representing the logical AND of ℓ independent instances of the Equality function on n bits (denoted EQ_n). That is, for

inputs $x = (x_1, \ldots, x_\ell)$ and $y = (y_1, \ldots, y_\ell)$ where each $x_i, y_i \in \{0,1\}^n$, the function is defined as:

$$f(x,y) = \bigwedge_{i=1}^{\ell} EQ_n(x_i, y_i) = \begin{cases} 1 & \text{if } x_i = y_i \text{ for all } i \in \{1, \ldots, \ell\} \\ 0 & \text{otherwise.} \end{cases}$$

To prove the lower bound $D(f) \geq \ell \cdot n$, we employ the fooling set technique. Recall that if a function g possesses a fooling set S of size $|S|$, then the deterministic communication complexity satisfies $D(g) \geq \log_2 |S|$. A set $S \subseteq \mathcal{X} \times \mathcal{Y}$ is a 1-fooling set for g if: 1. For all $(u,v) \in S$, $g(u,v) = 1$. 2. For all distinct pairs $(u,v), (u',v') \in S$, either $g(u,v') = 0$ or $g(u',v) = 0$.

First, consider the single instance EQ_n. We construct a fooling set S_0 for EQ_n by taking the set of identical input pairs:

$$S_0 = \{(w,w) \mid w \in \{0,1\}^n\}.$$

The size of this set is $|S_0| = 2^n$. It is a valid fooling set because for any $(w,w) \in S_0$, $EQ_n(w,w) = 1$, and for any distinct $(w,w), (w',w') \in S_0$, we have $w \neq w'$, which implies $EQ_n(w,w') = 0$.

Following the result of Exercise 4.12 regarding the composition of fooling sets, we construct a fooling set S for the composed function f by taking the Cartesian product of the fooling sets for the sub-functions. Let $S = S_0^\ell = S_0 \times S_0 \times \cdots \times S_0$. An element of S is a pair of sequences $(\vec{x}, \vec{x})$ where $\vec{x} = (w_1, \ldots, w_\ell)$ and each $w_i \in \{0,1\}^n$.

The size of the set S is:

$$|S| = |S_0|^\ell = (2^n)^\ell = 2^{\ell \cdot n}.$$

We verify that S is indeed a fooling set for f: 1. For any $(\vec{x}, \vec{x}) \in S$, we have $x_i = x_i$ for all i, so $f(\vec{x}, \vec{x}) = 1$. 2. Consider two distinct elements $(\vec{x}, \vec{x})$ and $(\vec{y}, \vec{y})$ in S. Since the elements are distinct, the vectors $\vec{x}$ and $\vec{y}$ must differ in at least one coordinate. Let $k \in \{1, \ldots, \ell\}$ be an index such that $x_k \neq y_k$. 3. Consider the crossed input pair $(\vec{x}, \vec{y})$. The output of the function depends on the component-wise equalities. Specifically, for the k-th component, we have inputs x_k and y_k. Since $x_k \neq y_k$, $EQ_n(x_k, y_k) = 0$. 4. Since f is the conjunction of the component functions, $f(\vec{x}, \vec{y}) = 0$. (Similarly, $f(\vec{y}, \vec{x}) = 0$).

Thus, S is a valid fooling set for f of size $2^{\ell \cdot n}$. The deterministic communication complexity is bounded by the logarithm of the fooling set size:

$$D(\varepsilon\alpha^\ell) \geq \log_2 |S| = \log_2(2^{\ell \cdot n}) = \ell \cdot n.$$

□

F.10.5 Grok

Proof. The function $\varepsilon\alpha^\ell$ is the ℓ-fold AND-composition of the n-bit equality function α, that is, on inputs $(u_1, \ldots, u_\ell, v_1, \ldots, v_\ell) \in (\{0,1\}^n)^\ell \times (\{0,1\}^n)^\ell$, we have $\varepsilon\alpha^\ell(u,v) = 1$ if and only if $u_i = v_i$ for all $i = 1, \ldots, \ell$.

First, we prove the lower bound using a fooling set argument. Consider the set

$$S = \left\{ \big((z_1, \ldots, z_\ell), (z_1, \ldots, z_\ell)\big) \;\middle|\; z_i \in \{0,1\}^n \; \forall\, i = 1, \ldots, \ell \right\}.$$

For every element of S, we have $\varepsilon\alpha^\ell = 1$. The size of S is $2^{\ell n}$. Now take two distinct elements of S, corresponding to strings $z = (z_1, \ldots, z_\ell)$ and $z' = (z'_1, \ldots, z'_\ell)$ with $z \neq z'$. There exists some i such that $z_i \neq z'_i$. Then the crossed input with Alice's input z and Bob's input z' satisfies $\varepsilon\alpha^\ell(z, z') = 0$, since the i-th blocks differ. Thus, S is a 1-fooling set of size $2^{\ell n}$.

In any deterministic protocol of communication cost c, the input matrix is partitioned into at most 2^c monochromatic rectangles. Since no monochromatic 1-rectangle can contain two distinct points of S (as that would force the crossed input to be 1 by the rectangle property, contradicting the fooling property), the number of 1-rectangles is at least $|S| = 2^{\ell n}$. Therefore, $2^c \geq 2^{\ell n}$, so $c \geq \ell n$. Hence, $D(\varepsilon\alpha^\ell) \geq \ell n$.

Alternatively, using Exercise 4.12: the n-bit equality function α has a 1-fooling set of size 2^n (the diagonal entries). Exercise 4.12 implies that the ℓ-fold AND-composition of a function with a fooling set of size m has a fooling set of size m^ℓ. Applying this to α yields a fooling set of size $(2^n)^\ell = 2^{\ell n}$ for $\varepsilon\alpha^\ell$. The same argument as above then gives $D(\varepsilon\alpha^\ell) \geq \ell n$. □

F.11 Problem 11

F.11.1 Problem Statement

Theorem F.11. *Let f be a Boolean function and let M_f be the matrix associated with f. If t is the number of different rows in M_f, then $D^1(f) = \log_2 t + 1$.*

F.11.2 GPT

Proof. Let f be a Boolean function and let M_f be the matrix associated with f. Denote by t the number of distinct rows in M_f. We aim to prove that $D^1(f) = \log_2 t + 1$.

First, recall the definition of the matrix M_f. The matrix M_f is the truth table of the Boolean function f, where each row corresponds to an input assignment of the function f, and the columns represent the bits of the input and the output of f. The number of rows in M_f is equal to the number of possible input assignments to f. If f has n variables, then the number of rows in M_f is 2^n.

Next, we observe that $D^1(f)$, the decision tree complexity of f, is the minimum number of queries needed to determine the value of f for an arbitrary input. In other words, it is the height of the smallest decision tree that computes f.

Since the matrix M_f contains t distinct rows, it follows that the number of distinct behaviors that f can exhibit over its inputs is t. Each distinct row in M_f represents a unique combination of inputs and corresponding output of f.

To construct a decision tree that computes f, we must distinguish between these t different rows. At each decision node of the tree, we must make a query that helps us identify which row of M_f we are currently observing. The minimum number of queries needed to distinguish between t distinct rows is $\lceil \log_2 t \rceil$, because each query provides a binary decision, effectively halving the set of possible rows.

Thus, the depth of the decision tree is at least $\lceil \log_2 t \rceil$. However, since $D^1(f)$ represents the decision tree complexity, we must also account for the case where all distinct rows are needed. The smallest possible depth of the decision tree is given by the formula $D^1(f) = \log_2 t + 1$, as this corresponds to the number of queries required to distinguish between all t distinct rows.

Therefore, we have shown that $D^1(f) = \log_2 t + 1$. □

F.11.3 Claude

Proof. We prove that $D^1(f) = \log_2 t + 1$, where $D^1(f)$ denotes the deterministic communication complexity of f with one-sided error and t is the number of distinct rows in the matrix M_f associated with the Boolean function f.

First, we establish the upper bound $D^1(f) \leq \log_2 t + 1$. Consider a protocol where Alice, who holds input x, determines which of the t distinct row types corresponds to her input. Since there are t distinct rows, Alice can encode her row type using $\lceil \log_2 t \rceil$ bits. Alice sends this encoding to Bob, who holds input y. Bob then looks up the value $f(x, y)$ by checking the entry in the transmitted row type at column y. This requires at most $\log_2 t + 1$ bits of communication (the additional bit accounts for the ceiling function and protocol overhead). Thus $D^1(f) \leq \log_2 t + 1$.

Next, we establish the lower bound $D^1(f) \geq \log_2 t + 1$. Suppose there exists a deterministic protocol computing f with communication cost $c < \log_2 t + 1$. In any deterministic protocol, the communication transcript partitions the input space. Specifically, for each possible transcript π, let R_π denote the set of rows (Alice's inputs) consistent with transcript π.

Since the protocol is deterministic and correct, all rows in R_π must agree on every column where the protocol outputs a value. However, if $c < \log_2 t + 1$, then there are at most $2^c < 2^{\log_2 t + 1} = 2t$ possible transcripts.

By the pigeonhole principle, since there are t distinct rows and fewer than $2t$ transcripts, but we need each distinct row to potentially determine a different function behavior, we require that the protocol can distinguish between all t distinct row types. A deterministic protocol with c bits of communication can create at most 2^c different transcripts. To distinguish t distinct rows, we need $2^c \geq t$, which implies $c \geq \log_2 t$.

More precisely, since each distinct row in M_f represents a different function from Bob's input space to $\{0, 1\}$, and the protocol must correctly compute $f(x, y)$ for all inputs, Alice must communicate enough

information to identify which of the t row types she holds. This requires at least $\log_2 t$ bits. The additional $+1$ term accounts for the fact that communication complexity is measured in bits sent, and the protocol structure requires at least $\lceil \log_2 t \rceil$ bits, which satisfies $\lceil \log_2 t \rceil \leq \log_2 t + 1$ with equality achieved for appropriate values of t.

Combining both bounds, we conclude that $D^1(f) = \log_2 t + 1$. □

F.11.4 Gemini

<think>**Defining the Problem's Scope**

I'm currently focused on the problem's scope. I am meticulously analyzing the request to ensure a comprehensive understanding. The goal is to identify the crucial components of the theorem statement presented in LaTeX. I'm taking notes on the role, input, and desired output for effective execution.

Revising Key Definitions

I'm now refining the definitions required for the proof. I am re-evaluating the definition of $D^1(f)$ to ensure it aligns perfectly with the theorem's context. I'm taking care to clarify the relationship between f, M_f, t, and the communication complexity to ensure logical consistency. The goal is a precise and readily understandable formalization.

Considering Proof Strategy

I'm now contemplating how to construct the proof. The goal is to prove $D^1(f) = \log_2 t + 1$. Given the definitions and context, a direct approach focusing on the distinct rows in M_f seems most promising. I need to figure out why the $+1$ is there and whether there's an underlying assumption.

Examining Potential Interpretations

I'm now diving deeper into possible interpretations of the theorem statement. I am questioning whether the "+ 1" implies something special about the encoding scheme. My notes outline two paths, a base result and a modified result. I'm focusing on the distinct rows in M_f and how Alice communicates them to Bob.

Examining Proof Implications

I'm now grappling with the implications of the $+1$ in the theorem. I'm exploring prefix-free codes, and other nuances related to the theorem statement. Since the theorem isn't fully standard, my next step is to examine the specific result or context from which it originates, to align the proof strategy more closely. I suspect this may be related to the encoding scheme or a specific definition of D^1.

Evaluating the Theorem's Basis

I'm now carefully examining the given theorem statement, specifically the $+1$ component. I'm exploring the possibility of a connection to decision tree depths, termination bits, or the requirements of prefix-free encoding. I'm revisiting core concepts to ensure a proof that precisely matches the prompt.

Refining Proof Construction

I'm now formulating the structure of the proof. I'm choosing a direct approach, showing Alice's message length is indeed $\lceil \log_2 t \rceil$ and adding a final bit, to give $\log_2 t + 1$. My plan is to formalize the encoding and decoding steps, followed by an explanation of the $+1$ bit (possibly related to termination or a specific code construction).

Examining Theorem Implications

I'm now investigating how the $+1$ arises in the theorem, and I am leaning towards exploring the nuances of this addition. The goal is to either justify the precise statement or prove a close variant. My hypothesis is that the $+1$ might represent an encoding scheme requirement, perhaps related to termination or a specific code construction.

Addressing Encoding and Overhead

I'm now contemplating how to handle the $+1$ in the theorem, and I must acknowledge the discrepancy between the prompt and the standard result. I hypothesize that this could relate to prefix-free coding or the theorem source and its specific definitions. My approach involves proving the standard result and then providing insights into the possible origins of the extra bit.

Structuring the Proof's Body

I am now structuring the proof, aiming to demonstrate Alice's message length. I'll show that Alice only needs to send an index representing the row, needing $\lceil \log_2 t \rceil$ bits. This will form the core of my argument, laying the foundation to examine the implications of the $+1$ in the theorem.

Formulating the Core Argument

I am now focusing on the fundamental argument, using the distinct rows in M_f. Alice needs to send Bob an index, i, to identify her row, with each index corresponding to one of the t different rows. The key is in how many bits are required for the encoding scheme used to represent this index.

Formulating the Core Argument

I am now focusing on the essential argument, using the matrix M_f's distinct rows. Alice must send Bob an index, i, representing her row, with each i uniquely identifying one of t distinct rows. The next step is evaluating the information each of these distinct rows conveys.

Structuring the Proof's Body

I'm now structuring the proof, emphasizing the number of distinct rows. Alice sends Bob the index of the row, and the number of distinct rows determines the number of bits needed. The final touches will focus on the role of the $+1$ in the theorem, where I'll connect to encoding schemes or upper bounds.

Illustrating the Core Logic

I am now focusing on the central argument: how Alice can efficiently transmit information. Since there are t distinct rows, it is crucial to find the simplest communication strategy to transmit the distinct row index. My strategy centers on the distinct rows in M_f.

Formulating Proof Conclusion

I'm now determining how to conclude the proof, with the goal of demonstrating the validity of the theorem as expressed. My strategy is to clearly showcase that Alice needs $\lceil \log_2 t \rceil$ bits. I will connect this to the bounds and assumptions in the theorem.

Formulating Proof Conclusion

I'm now structuring my conclusion to the proof. My goal is to precisely represent the communication complexity concerning the distinct rows. I'll summarize the upper and lower bounds derived to clearly establish the value of $D^1(f)$.

Constructing the Core Proof

I am now focusing on the essential core of the proof. The goal is to establish the upper and lower bounds on the number of bits Alice needs to send. The approach will concentrate on analyzing the message lengths needed for communication.

Formulating a Concise Proof

I am now focusing on the essential core of the proof's presentation. I'm choosing to define the upper and lower bounds on the number of bits required in Alice's message, which directly relates to the concept of communication complexity. I will carefully formulate my argument to cover both the distinct rows and the implications of Alice's communication scheme.

Analyzing the Core Theorem

I'm now closely analyzing the theorem's conclusion, that $D^1(f) = \log_2 t + 1$. I am carefully considering how this precise statement can be derived, paying special attention to the role of the '+1' in the context of one-way communication complexity. I will prove $D^1(f) = \lceil \log_2 t \rceil$ and note that it relates.

Formulating the Final Proof

I am now putting together the complete proof. My approach leverages the number of distinct rows and the need for Alice to communicate. I'm focusing on ensuring each distinct row can be indexed uniquely and efficiently. The goal is to provide a comprehensive and accurate argument to define the number of bits.

Formulating Final Conclusions

I'm now writing my conclusion, focusing on the bounds for $D^1(f)$. I will emphasize the role of distinct rows and the implications of encoding strategies on communication complexity. My aim is to deliver a complete and accurate argument.

Evaluating the Final Output

I'm now carefully reviewing the final LaTeX code I generated to ensure its adherence to the instructions. My strategy focuses on justifying the standard result, accounting for the theorem's specific formulation. The goal is a concise and complete proof.

Proof Structuring Completion

I am now putting together the proof's final elements and conclusion. The goal is to clearly present all the aspects of the main idea, emphasizing the essential components for a complete result. The concluding summary needs to address the $+1$.

Proof Review and Polish

I am now doing a final review of the generated proof and making the final adjustments to the text. The goal is to ensure completeness, clarity, and compliance with the prompt's formatting and output requirements. I've re-read the theorem statement to make sure the proof adequately covers the theorem's claims.

Addressing Proof Conclusion

I'm now focusing on the final step, providing the conclusion. I am constructing a concise summary of the upper and lower bounds to solidify the result and address the specified theorem. While the standard result is $\lceil \log_2 t \rceil$, I will attempt to relate to the provided statement in the theorem.

</think>

Proof. Let $f : X \times Y \to \{0,1\}$ be a Boolean function and let M_f be the communication matrix associated with f, where the entry (x, y) corresponds to $f(x, y)$. Let R be the set of distinct rows of M_f, and let $t = |R|$ be the number of different rows. We analyze the deterministic one-way communication complexity $D^1(f)$, where Alice sends a single message to Bob, who then computes the function value.

First, we establish an upper bound. Alice can partition the set of inputs X into t equivalence classes, where x and x' are in the same class if and only if row x and row x' in M_f are identical. Alice and Bob agree on a canonical ordering of the t distinct row vectors $r_1, \ldots, r_t$. Given an input x, Alice identifies the index $k \in \{1, \ldots, t\}$ such that the row corresponding to x is r_k. She sends the index k to Bob using a fixed-length binary encoding. This requires $\lceil \log_2 t \rceil$ bits. Upon receiving k, Bob retrieves the row vector r_k and outputs the value in the column corresponding to his input y, which is $r_k(y) = f(x, y)$. Since this protocol is valid, we have $D^1(f) \leq \lceil \log_2 t \rceil$.

Next, we establish a lower bound. Let Π be any deterministic one-way protocol for f with worst-case cost c. In this protocol, Alice sends a message $m(x)$ to Bob. The correctness of the protocol implies that the message $m(x)$ combined with Bob's input y uniquely determines $f(x, y)$. Suppose there exist two inputs $x_1, x_2 \in X$ such that their corresponding rows in M_f are distinct, yet Alice sends the same message $m(x_1) = m(x_2)$. Since the rows are distinct, there exists a column $y_0 \in Y$ such that $f(x_1, y_0) \neq f(x_2, y_0)$. However, because Bob receives the same message for both x_1 and x_2, his computation depends only on the message and y_0, leading to the same output for both cases. This contradicts the fact that $f(x_1, y_0) \neq f(x_2, y_0)$. Therefore, Alice must map inputs corresponding to distinct rows to distinct messages. Since there are t distinct rows, the set of possible messages must have size at least t. The length of the messages c must satisfy $2^c \geq t$, which implies $c \geq \log_2 t$. Since c must be an integer, $c \geq \lceil \log_2 t \rceil$. Thus, $D^1(f) \geq \lceil \log_2 t \rceil$.

Combining the upper and lower bounds, we obtain $D^1(f) = \lceil \log_2 t \rceil$. □

F.11.5 Grok

Proof. The communication matrix M_f has rows corresponding to Alice's inputs and columns to Bob's inputs, with the (x, y)-entry equal to $f(x, y)$.

Let t be the number of distinct rows in M_f, and fix an enumeration $r_1, \ldots, r_t$ of these distinct rows.

The players agree on this enumeration in advance.

For the upper bound $D^1(f) \leq \log_2 t + 1$:

Let $k = \lfloor \log_2 t \rfloor + 1$.

Then $2^k \geq t$.

The players agree on t distinct k-bit strings $s_1, \ldots, s_t$.

On input x, Alice determines the unique index i such that the row corresponding to x is r_i, and sends the string s_i (of length k).

Bob receives s, determines the corresponding i, and outputs the entry of r_i in position y (his input).

This protocol correctly computes f, with communication cost $k \leq \log_2 t + 1$.

Thus, $D^1(f) \leq \log_2 t + 1$.

For the lower bound $D^1(f) \geq \log_2 t$:

Suppose there is a deterministic one-way protocol with cost $c < \log_2 t$.

Then Alice sends at most c bits, so there are at most $2^c < t$ possible messages.

The map from Alice's inputs x to the message sent on x partitions Alice's inputs into at most $2^c < t$ parts.

Each part must consist of inputs with identical rows in M_f (otherwise, Bob could not output correctly for all x in the part on a y where the rows differ).

Thus, each part is contained in one row-type equivalence class.

Since there are t distinct row types, and fewer than t parts, by the pigeonhole principle, at least one row type has no inputs assigned to any part, or more precisely, the parts cover at most $2^c < t$ row types.

However, all row types are realized (there exists at least one x for each), so all t row types must be covered by the parts, a contradiction.

Therefore, no such protocol exists with cost $c < \log_2 t$, so $D^1(f) \geq \log_2 t$.

Combining the bounds, $\log_2 t \leq D^1(f) \leq \log_2 t + 1$.

□

F.12 Problem 12

F.12.1 Problem Statement

Theorem F.12 (4.20)**.** *Prove that* $R^1(\textsc{Index}) = \Theta(n)$ *and that* $R^{1,Bob}(\textsc{Index}) = \Theta(\log n)$.

F.12.2 GPT

Proof. We are tasked with proving that $R^1(\textsc{Index}) = \Theta(n)$ and $R^{1,\text{Bob}}(\textsc{Index}) = \Theta(\log n)$.

Part 1: Proof that $R^1(\textbf{INDEX}) = \Theta(n)$

Let $R^1(\textsc{Index})$ represent the runtime complexity of the index operation. To prove that $R^1(\textsc{Index}) = \Theta(n)$, we need to show that there exist constants $c_1, c_2 > 0$ and n_0 such that for all $n \geq n_0$,

$$c_1 n \leq R^1(\textsc{Index}) \leq c_2 n.$$

First, observe that the index operation typically involves accessing or processing elements in a list, array, or similar data structure. If the operation involves iterating through all n elements, the runtime will be proportional to n. Specifically, there exists a constant c_1 such that for sufficiently large n:

$$R^1(\textsc{Index}) \geq c_1 n.$$

On the other hand, the index operation cannot take more than a constant factor times n steps, since it cannot exceed the cost of performing a linear scan. Thus, there exists a constant c_2 such that for all $n \geq n_0$:

$$R^1(\textsc{Index}) \leq c_2 n.$$

Therefore, we conclude that $R^1(\textsc{Index}) = \Theta(n)$.

Part 2: Proof that $R^{1,\textbf{Bob}}(\textbf{INDEX}) = \Theta(\log n)$

Next, we prove that $R^{1,\text{Bob}}(\textsc{Index}) = \Theta(\log n)$. Let $R^{1,\text{Bob}}(\textsc{Index})$ represent the runtime complexity of the index operation under Bob's strategy. Typically, Bob's strategy implies the use of a binary search or a similar logarithmic time algorithm to find an index. To prove that $R^{1,\text{Bob}}(\textsc{Index}) = \Theta(\log n)$, we need to show that there exist constants $c_3, c_4 > 0$ and n_0 such that for all $n \geq n_0$,

$$c_3 \log n \leq R^{1,\text{Bob}}(\textsc{Index}) \leq c_4 \log n.$$

Binary search, when applied to a sorted list of size n, requires at most $\log_2 n$ comparisons, and thus the runtime is proportional to $\log n$. There exists a constant c_3 such that for sufficiently large n:

$$R^{1,\text{Bob}}(\textsc{Index}) \geq c_3 \log n.$$

Since binary search cannot take more than a constant factor times $\log n$ steps, there exists a constant c_4 such that for all $n \geq n_0$:

$$R^{1,\text{Bob}}(\text{INDEX}) \leq c_4 \log n.$$

Therefore, we conclude that $R^{1,\text{Bob}}(\text{INDEX}) = \Theta(\log n)$.

Thus, we have proven both parts of the exercise:

$$R^1(\text{INDEX}) = \Theta(n) \quad \text{and} \quad R^{1,\text{Bob}}(\text{INDEX}) = \Theta(\log n).$$

□

F.12.3 Claude

Proof. We prove both parts separately.

Part 1: $R^1(\textbf{INDEX}) = \Theta(n)$.

Recall that in the INDEX problem, Alice holds an n-bit string $x \in \{0,1\}^n$ and Bob holds an index $i \in [n]$, and Bob must output x_i. Here $R^1(\text{INDEX})$ denotes the one-way communication complexity where only Alice sends a message to Bob.

Upper bound: Alice can send her entire input x to Bob, who then outputs x_i. This requires n bits of communication, so $R^1(\text{INDEX}) \leq n$.

Lower bound: We show that $R^1(\text{INDEX}) \geq n$ by a counting argument. Consider any deterministic one-way protocol π where Alice sends a message $m(x)$ based on her input x. For the protocol to be correct, Bob must be able to determine x_i for every $i \in [n]$ given only $m(x)$ and i.

Suppose Alice's message has length $\ell < n$. Then there are at most $2^\ell < 2^n$ possible messages. By the pigeonhole principle, there exist two distinct inputs $x, x' \in \{0,1\}^n$ such that $m(x) = m(x')$. Since $x \neq x'$, there exists some index $j \in [n]$ where $x_j \neq x'_j$.

When Bob receives message $m(x) = m(x')$ and index j, he cannot distinguish whether Alice's input is x or x'. Therefore, Bob cannot correctly output both x_j and x'_j, which means the protocol fails on at least one of these inputs. This contradicts the correctness of the protocol.

Therefore, $R^1(\text{INDEX}) \geq n$, and we conclude $R^1(\text{INDEX}) = \Theta(n)$.

Part 2: $R^{1,\textbf{Bob}}(\textbf{INDEX}) = \Theta(\log n)$.

Here $R^{1,\text{Bob}}(\text{INDEX})$ denotes the one-way communication complexity where only Bob sends a message to Alice, and Alice must output the answer.

Upper bound: Bob can simply send his index i to Alice, who then outputs x_i. Since $i \in [n]$, this requires $\lceil \log_2 n \rceil$ bits, so $R^{1,\text{Bob}}(\text{INDEX}) = O(\log n)$.

Lower bound: We show that $R^{1,\text{Bob}}(\text{INDEX}) \geq \log_2 n$ by an information-theoretic argument. Consider any deterministic protocol where Bob sends a message $m(i)$ based on his index $i \in [n]$. For the protocol to be correct, Alice must be able to determine x_i for every input x and every index i given only x and $m(i)$.

Suppose Bob's message has length $\ell < \log_2 n$. Then there are at most $2^\ell < n$ possible messages. By the pigeonhole principle, there exist two distinct indices $i, j \in [n]$ with $i \neq j$ such that $m(i) = m(j)$.

Consider the input $x \in \{0,1\}^n$ where $x_i = 0$ and $x_j = 1$. When Alice receives message $m(i) = m(j)$, she cannot distinguish whether Bob's index is i or j. If Bob's index is i, Alice should output $x_i = 0$, but if Bob's index is j, Alice should output $x_j = 1$. Since Alice must give a single output, the protocol fails on at least one of these cases.

Therefore, $R^{1,\text{Bob}}(\text{INDEX}) \geq \log_2 n$, and we conclude $R^{1,\text{Bob}}(\text{INDEX}) = \Theta(\log n)$. □

F.12.4 Gemini

Proof. We first prove the one-way communication complexity for the INDEX function, $R^1(\text{INDEX}) = \Theta(n)$. The upper bound is trivial, as Alice can send the entire input x to Bob, enabling Bob to compute x_i directly, which requires n bits. We now concentrate on establishing the lower bound. Alice has $x \in \{0,1\}^n$, and Bob has $i \in [n]$, and Bob must output x_i. Assume Alice sends a message $M(x)$ of length C bits. Bob, upon receiving M and knowing i, outputs $O(M, i)$. The error probability is

bounded by $\Pr[O(M,i) \neq x_i] \leq \delta$. We will use the information-theoretic approach. Consider the joint distribution μ where $X \sim U(\{0,1\}^n)$ and $I \sim U([n])$. Let Y be the message sent by Alice. We have that $I(X;Y) \geq I(X;Y|I)$. We know that $H(X_i|M,I) \leq H(\delta)$. Hence, $I(X_I;Y|I) = H(X_I|I) - H(X_I|I,Y) \geq 1 - H(\delta)$. Also $I(X;Y) = \sum_{i=1}^{n} I(X_i;Y)$. Because X_i is independent for each i. We can see that $I(X;M) = \sum_{j=1}^{n} I(X_j;M) \geq \sum_{j=1}^{n}(1 - H(\delta)) = n(1 - H(\delta))$. We know that $|M| \geq I(X;M)$. We get $|M| = \Omega(n)$.

Now, we consider $R^{1,\text{Bob}}(\text{INDEX})$. The upper bound is straightforward: Bob sends the index i requiring $\lceil \log n \rceil$ bits, and Alice retrieves the correct x_i using her input. Therefore, $R^{1,\text{Bob}}(\text{INDEX}) \leq \lceil \log n \rceil$. For the lower bound, let M be the message sent from Bob to Alice. Alice receives M and input x. Bob has input i, and the output must be x_i. If Bob sends C bits, there are 2^C possible messages. By the standard argument, we know that if Bob sends less than $\log n$ bits. We get $\Omega(\log n)$ bits. □

F.12.5 Grok

Proof. The indexing function $\text{INDEX} : \{0,1\}^n \times [n] \to \{0,1\}$ is defined by $\text{INDEX}(x,i) = x_i$, where Alice holds $x \in \{0,1\}^n$ and Bob holds $i \in [n]$.

We first prove $R^{1,\text{Bob}}(\text{INDEX}) = \Theta(\log n)$.

For the upper bound, Bob sends the $\lceil \log_2 n \rceil$ bits encoding i to Alice, who outputs x_i. The protocol has zero error and communication $O(\log n)$.

For the lower bound, the $2^n \times n$ communication matrix has n pairwise distinct columns, since for $i \neq j$ there exists x with $x_i \neq x_j$. By the symmetric analogue of the deterministic one-way complexity being $\Theta(\log_2 t)$ for t distinct rows, the deterministic one-way complexity from Bob to Alice is $\Theta(\log n)$. Randomization does not asymptotically reduce the complexity here, as message distributions shared across distinct indices cause conflicts on inputs where the corresponding bits differ, leading to error exceeding $1/3$ for some inputs.

Thus, $R^{1,\text{Bob}}(\text{INDEX}) = \Theta(\log n)$.

Next, we prove $R^{1}(\text{INDEX}) = \Theta(n)$.

For the upper bound, Alice sends the n bits of x to Bob, who outputs the i-th bit. The protocol has zero error and communication n.

For the lower bound, assume a public-coin bounded-error $(1/3)$ protocol with communication cost c. There are at most 2^c possible messages.

For fixed public randomness r, the protocol is deterministic: Alice sends $m(r,x)$, and Bob outputs a bit depending on r, m, and i. For each m and r, define $y_m^r \in \{0,1\}^n$ by $(y_m^r)_j =$ Bob's output on index j receiving m (with r). The codebook $C_r = \{y_m^r\}$ has size at most 2^c.

Alice selects some $y \in C_r$ depending on x and r.

Since error is at most $1/3$, for every x and i, $\Pr_r[(y)_i = x_i] \geq 2/3$. Thus, for every x, the expected (over r) agreement between x and the selected y is at least $(2/3)n$.

Even choosing the $y \in C_r$ maximizing agreement, the expected agreement is at most the expected max agreement, equivalent to $n - \mathbb{E}_r[\min_{y \in C_r} d_H(x,y)]$.

The requirement implies $\mathbb{E}_r[\min_{y \in C_r} d_H(x,y)] \leq (1/3)n$ for every x.

The volume of a Hamming ball of radius $(1/3)n$ is at most $2^{h(1/3)n}$, where $h(p) = -p\log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function and $h(1/3) < 1$. Let $\delta = 1 - h(1/3) > 0$.

For any fixed r, the number of x with $\min d_H(x, C_r) \leq (1/3)n$ is at most $2^c \cdot 2^{h(1/3)n + o(n)}$. If $c \leq (\delta/2)n$, this is $o(2^n)$, so the fraction of well-covered x (distance $\leq (1/3)n$ to C_r) is $2^{c + h(1/3)n - n + o(n)} = 2^{-\Omega(n)}$.

Thus, $\Pr_r[x \text{ is well-covered by } C_r] = 2^{-\Omega(n)}$.

For r where x is not well-covered, $\min d_H > (1/3)n$, so agreement $< (2/3)n$ (strictly, as distances are integer).

The expected agreement for any x is at most $2^{-\Omega(n)} \cdot n + (1 - 2^{-\Omega(n)}) \cdot ((2/3)n - 1) < (2/3)n$ for large n, contradicting the requirement.

Hence, $c = \Omega(n)$, so $R^{1}(\text{INDEX}) = \Theta(n)$. □

F.13 Problem 13

F.13.1 Problem Statement

Theorem F.13 (4.21)**.** *Show that* $D(f) \geq \log_2 D^1(f)$ *and also* $R(f) \geq \log_2 R^1(f)$.

F.13.2 GPT

Proof. We are tasked with proving two inequalities:

1. $D(f) \geq \log_2 D^1(f)$ 2. $R(f) \geq \log_2 R^1(f)$

Proof of $D(f) \geq \log_2 D^1(f)$

Let $D(f)$ denote the dimension of the function f and $D^1(f)$ denote the first derivative dimension of f. By the definition of the dimension of a function, we know that:

$$D(f) = \sup \{d : f \text{ has a d-dimensional fractal structure}\}$$

Similarly, for the first derivative, we have:

$$D^1(f) = \sup \left\{d : f' \text{ has a d-dimensional fractal structure}\right\}$$

By the chain rule, the derivative function f' can only reduce the fractal dimension of the original function f. Hence, the dimension of the derivative cannot exceed the dimension of the original function:

$$D(f') \leq D(f)$$

Since $D^1(f)$ represents the dimension of the first derivative, it follows that:

$$D^1(f) \leq D(f)$$

Thus, we conclude that:

$$D(f) \geq D^1(f)$$

Furthermore, we know that the dimension of a function is related to the logarithm of the number of bits needed to represent it. Hence, we have:

$$D^1(f) \leq D(f) \implies \log_2 D^1(f) \leq \log_2 D(f)$$

Thus, we obtain the desired inequality:

$$D(f) \geq \log_2 D^1(f)$$

Proof of $R(f) \geq \log_2 R^1(f)$

Let $R(f)$ denote the regularity of the function f, and $R^1(f)$ denote the regularity of its first derivative. The regularity of a function is a measure of how smooth or well-behaved the function is. For the first derivative, we can observe that differentiating a function typically reduces its regularity. Specifically, we have:

$$R(f') \leq R(f)$$

This inequality arises from the fact that the regularity of a function's derivative is typically less than or equal to the regularity of the function itself. Since $R^1(f)$ represents the regularity of the first derivative, it follows that:

$$R^1(f) \leq R(f)$$

As with the dimension case, the regularity is related to the logarithm of the representation, so we conclude:

$$R^1(f) \leq R(f) \implies \log_2 R^1(f) \leq \log_2 R(f)$$

Thus, we obtain the desired inequality:

$$R(f) \geq \log_2 R^1(f)$$

Therefore, both inequalities are proven:

$$D(f) \geq \log_2 D^1(f) \quad \text{and} \quad R(f) \geq \log_2 R^1(f)$$

□

F.13.3 Claude

Proof. We prove both inequalities separately.

Part 1: $D(f) \geq \log_2 D^1(f)$**.**

Let P be an optimal deterministic communication protocol for f with depth $D(f)$. We construct a corresponding 1-way deterministic protocol as follows: Alice sends a message encoding the leaf reached in the protocol tree after her first message, which requires at most $D(f)$ bits since the protocol tree has depth $D(f)$ and thus at most $2^{D(f)}$ leaves reachable after Alice's first message.

More formally, consider any deterministic protocol computing f with communication cost c. In the worst case, this protocol can distinguish between at most 2^c different transcripts. Since a 1-way protocol must have Alice send enough information for Bob to compute $f(x, y)$ for his input y, Alice must send a message that allows Bob to distinguish between all possible behaviors of f restricted to Alice's input.

For any 1-way deterministic protocol computing f, Alice must send at least $\log_2 D^1(f)$ bits, since there are $D^1(f)$ distinct messages Alice might need to send in an optimal 1-way protocol. Conversely, any deterministic protocol with communication $D(f)$ can simulate a 1-way protocol by having Alice send her entire view of an optimal protocol execution, which requires at most $D(f)$ bits.

Since $D^1(f)$ is the minimum number of bits required in any 1-way deterministic protocol, and any deterministic protocol with communication c can be converted to a 1-way protocol with communication at most c, we have that there exists a 1-way protocol using $D(f)$ bits. Therefore, $D^1(f) \leq 2^{D(f)}$, which gives us $\log_2 D^1(f) \leq D(f)$.

Part 2: $R(f) \geq \log_2 R^1(f)$**.**

The proof follows analogously for randomized protocols. Let P be an optimal randomized protocol for f with expected communication cost $R(f)$ that errs with probability at most $1/3$.

Any randomized protocol can be converted to a 1-way randomized protocol as follows: Alice uses her random bits and input to determine what message she would send in the interactive protocol, then sends a message encoding the leaf of the protocol tree (or a sufficient statistic thereof). This requires expected communication at most $R(f)$.

More precisely, for any randomized protocol with expected communication c that computes f with error probability at most $1/3$, we can construct a 1-way randomized protocol with expected communication at most c: Alice simulates the entire protocol tree using shared randomness and her input, then sends Bob sufficient information (at most c bits in expectation) for Bob to compute the output.

Since $R^1(f)$ is the minimum expected communication over all 1-way randomized protocols computing f with error at most $1/3$, and we can construct such a 1-way protocol using expected communication $R(f)$, we have $R^1(f) \leq 2^{R(f)}$.

Taking logarithms, we obtain $\log_2 R^1(f) \leq R(f)$, which completes the proof. □

F.13.4 Gemini

Proof. Let $D(f)$ be the deterministic communication complexity of f, and let $D^1(f)$ be the minimum number of monochromatic rectangles needed to partition $f^{-1}(1)$. Any deterministic protocol for f induces a partition of the input space into monochromatic rectangles. Consider a deterministic protocol with cost $D(f)$. The leaves of its corresponding protocol tree form a partition of the input space into monochromatic rectangles. Let $\mathcal{L}$ be the set of leaves in this tree, with $|\mathcal{L}| \leq 2^{D(f)}$. Let $\mathcal{L}_1 \subseteq \mathcal{L}$ be the leaves where the protocol outputs 1. Since the protocol correctly computes f, $\bigcup_{l \in \mathcal{L}_1} R_l = f^{-1}(1)$,

meaning the 1-leaves partition $f^{-1}(1)$. By definition, $D^1(f)$ is the minimal size of such a partition. Hence, $D^1(f) \leq |\mathcal{L}_1| \leq |\mathcal{L}| \leq 2^{D(f)}$, which implies $D(f) \geq \log_2 D^1(f)$.

Let $R(f)$ be the randomized communication complexity of f, and let $R^1(f)$ be the minimum value of the linear programming relaxation of the integer program that finds the minimum number of monochromatic rectangles partitioning $f^{-1}(1)$. In other words, $R^1(f)$ is the fractional partition number. A randomized protocol for f (with public coin) can be viewed as a distribution over deterministic protocols. Let P be a randomized protocol and let P_i be the deterministic protocols with probability p_i. Each deterministic protocol P_i defines a partition of the input space into monochromatic rectangles. For each rectangle R in the partition induced by P_i, let w_R be the sum of weights. Construct a fractional partition as follows: for each rectangle R in the partition induced by P_i, assign the weight $w_R = p_i$. Then, for any $(x, y) \in f^{-1}(1)$, the sum of weights of rectangles containing (x, y) equals 1. Therefore, this collection of rectangles forms a valid fractional partition of $f^{-1}(1)$, and the total weight of this fractional partition is at most $2^{R(f)}$. Since $R^1(f)$ is the minimum total weight, it follows that $R^1(f) \leq 2^{R(f)}$. Thus, $R(f) \geq \log_2 R^1(f)$. □

F.13.5 Grok

Proof. We prove the inequalities separately, beginning with the deterministic case.

Let $t(f)$ denote the number of distinct rows in the communication matrix M_f. It holds that

$$\log_2 t(f) \leq D^1(f) \leq \log_2 t(f) + 1.$$

We prove by induction on d that if $D(f) \leq d$, then $t(f) \leq 2^{2^d}$.

For the base case $d = 0$, there is no communication and f is constant, so $t(f) = 1 \leq 2^{2^0} = 2$.

Assume the statement holds for all functions with communication complexity strictly less than d, where $d \geq 1$. Consider a function f with $D(f) \leq d$ and an optimal protocol Π of cost at most d.

The protocol begins with a message of length l ($0 \leq l \leq d$) sent by either Alice or Bob. (A message of length 0 corresponds to passing the turn.)

If Alice sends the first message of length l, there are at most 2^l possible messages, inducing at most 2^l disjoint subsets of Alice's inputs. For each such subset, the continuation is a subprotocol of cost at most $d - l$ on the corresponding submatrix (with the full set of Bob's inputs). By the induction hypothesis, each submatrix has at most $2^{2^{d-l}}$ distinct rows. Since the subsets are disjoint,

$$t(f) \leq 2^l \cdot 2^{2^{d-l}} = 2^{l+2^{d-l}} \leq 2^{2^d},$$

as $l + 2^{d-l} \leq 2^d$.

If Bob sends the first message of length l, there are at most 2^l possible messages, inducing at most 2^l disjoint subsets of Bob's inputs. For each such subset, the continuation is a subprotocol of cost at most $d - l$ on the corresponding submatrix (with the full set of Alice's inputs). By the induction hypothesis, the rows restricted to each subset of columns number at most $2^{2^{d-l}}$ distinct possibilities. Since the full rows are concatenations over the disjoint column subsets,

$$t(f) \leq \left(2^{2^{d-l}}\right)^{2^l} = 2^{2^{d-l} \cdot 2^l} = 2^{2^d}.$$

In both cases, $t(f) \leq 2^{2^d}$. By induction, the bound holds whenever $D(f) \leq d$. Thus,

$$D^1(f) \leq \log_2 t(f) + 1 \leq 2^{D(f)} + 1,$$

so

$$D(f) \geq \log_2 D^1(f) + O(1).$$

For the randomized case, the proof follows a similar structural argument on the protocol tree, but accounts for bounded error via a union bound over inputs or a probabilistic covering argument with Hamming balls (as in the indexing function, where tightness is achieved). The effective number of

distinguishable (approximate) row types remains doubly exponentially bounded in the communication cost, yielding

$$R(f) \geq \log_2 R^1(f) + O(1).$$

The additive constants are absorbed in the stated inequalities, which hold asymptotically (and exactly up to small additives or redefinitions of the complexities).

□

F.14 Problem 14

F.14.1 Problem Statement

Theorem F.14 (Exercise 4.22)**.** *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be any function. Let $D^{\|}(f)$ denote the deterministic communication complexity of computing the function f using simultaneous protocols and $R^{\|}(f)$ denote the randomized communication complexity of computing the function f using simultaneous protocols. Then*

$$D^{\|}(f) = D^{\uparrow}(f) + D^{\uparrow,Bob}(f).$$

F.14.2 GPT

Proof. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function, and let $D^{\|}(f)$, $D^{\uparrow}(f)$, and $D^{\uparrow,\text{Bob}}(f)$ be the deterministic communication complexities of computing f under simultaneous protocols for Alice and Bob. Our goal is to prove that

$$D^{\|}(f) = D^{\uparrow}(f) + D^{\uparrow,\text{Bob}}(f).$$

Step 1: Understanding the Terms We begin by recalling the definitions of the complexities involved: - $D^{\|}(f)$ is the deterministic communication complexity of f when Alice and Bob can communicate simultaneously, meaning that they can send their messages to each other independently and in parallel. - $D^{\uparrow}(f)$ is the deterministic communication complexity of f when Alice sends her message first and Bob sends his message afterward. - $D^{\uparrow,\text{Bob}}(f)$ is the deterministic communication complexity of f when Bob sends his message first and Alice sends hers afterward.

Thus, $D^{\|}(f)$ represents the situation where Alice and Bob can simultaneously send messages, and $D^{\uparrow}(f)$ and $D^{\uparrow,\text{Bob}}(f)$ correspond to the scenarios where one party must wait for the other to send their message first.

Step 2: Upper Bound on $D^{\|}(f)$ Since $D^{\uparrow}(f)$ and $D^{\uparrow,\text{Bob}}(f)$ correspond to deterministic communication complexities of sequential protocols, it is clear that:

$$D^{\|}(f) \leq D^{\uparrow}(f) + D^{\uparrow,\text{Bob}}(f),$$

because in a simultaneous protocol, the messages sent by Alice and Bob can be seen as independent, while in the sequential protocols, one message must be sent after the other. Hence, the total number of communication steps in a simultaneous protocol cannot exceed the sum of the communication steps in the two sequential protocols.

Step 3: Lower Bound on $D^{\|}(f)$ Next, we establish that the communication complexity in the simultaneous case must be at least the sum of the communication complexities in the sequential cases. The key observation is that, in the simultaneous setting, the parties cannot use any extra communication power to avoid the costs incurred by each party's individual communication sequence. In particular, Alice's message must carry at least as much information as the message in the $D^{\uparrow}(f)$ protocol, and Bob's message must carry at least as much information as the message in the $D^{\uparrow,\text{Bob}}(f)$ protocol. Hence, the total complexity in the simultaneous case is lower-bounded by the sum of the complexities in the sequential protocols:

$$D^{\|}(f) \geq D^{\uparrow}(f) + D^{\uparrow,\text{Bob}}(f).$$

Step 4: Conclusion Combining the upper bound and the lower bound, we conclude that

$$D^{\|}(f) = D^{\uparrow}(f) + D^{\uparrow,\text{Bob}}(f),$$

which completes the proof. □

F.14.3 Claude

Proof. We prove the equality $D^{\|}(f) = D^{\uparrow}(f) + D^{\uparrow,\text{Bob}}(f)$ by establishing both inequalities.

First direction: $D^{\|}(f) \leq D^{\uparrow}(f) + D^{\uparrow,\mathbf{Bob}}(f)$.

Let π be an optimal one-way protocol for f where Alice sends a message to Bob, using $D^{\uparrow}(f)$ bits of communication. Let π' be an optimal one-way protocol where Bob sends a message to Alice, using $D^{\uparrow,\text{Bob}}(f)$ bits of communication.

We construct a simultaneous protocol as follows: On input $(x, y) \in \{0,1\}^n \times \{0,1\}^n$, Alice sends the message she would send in protocol π on input x, and Bob sends the message he would send in protocol π' on input y. The referee receives both messages and can simulate both protocols. From the message received from Alice (which uses $D^{\uparrow}(f)$ bits), combined with Bob's input y, the referee can compute what Bob would output in protocol π, namely $f(x, y)$. Alternatively, from the message received from Bob (which uses $D^{\uparrow,\text{Bob}}(f)$ bits), combined with Alice's input x, the referee can compute what Alice would output in protocol π', namely $f(x, y)$.

The total communication is $D^{\uparrow}(f) + D^{\uparrow,\text{Bob}}(f)$ bits. Therefore, $D^{\|}(f) \leq D^{\uparrow}(f) + D^{\uparrow,\text{Bob}}(f)$.

Second direction: $D^{\|}(f) \geq D^{\uparrow}(f) + D^{\uparrow,\mathbf{Bob}}(f)$.

Let σ be an optimal simultaneous protocol for f using $D^{\|}(f)$ bits total. In this protocol, Alice sends a bits and Bob sends b bits, where $a + b = D^{\|}(f)$.

From the simultaneous protocol σ, we can construct a one-way protocol from Alice to Bob: Alice sends her a-bit message from σ. Bob, upon receiving this message and knowing his own input y, can compute what the referee would compute in protocol σ (since the referee's computation depends only on Alice's message, Bob's message, and the inputs, and Bob can generate his own message). Thus, Bob can compute $f(x, y)$. This shows that $D^{\uparrow}(f) \leq a$.

Similarly, we can construct a one-way protocol from Bob to Alice: Bob sends his b-bit message from σ. Alice, upon receiving this message and knowing her own input x, can compute what the referee would compute in protocol σ. Thus, Alice can compute $f(x, y)$. This shows that $D^{\uparrow,\text{Bob}}(f) \leq b$.

Therefore, $D^{\uparrow}(f) + D^{\uparrow,\text{Bob}}(f) \leq a + b = D^{\|}(f)$.

Combining both directions, we conclude that $D^{\|}(f) = D^{\uparrow}(f) + D^{\uparrow,\text{Bob}}(f)$. □

F.14.4 Gemini

<think>**Defining the Problem's Scope**

I'm focused on breaking down the initial request. My primary objective is to clarify the problem, identifying the core theorem regarding communication complexity. I am looking at its key elements. I need to fully understand its input and output to effectively approach the mathematical proof. The theorem is the driving factor.

Revisiting Key Definitions

Okay, I'm now zeroing in on the ambiguities in the notation. My goal is to ensure the equation even makes sense. The existing definitions of one-way communication don't neatly fit the simultaneous context. I'm going to look to understand the equation within the simultaneous message context better, which will allow me to formulate a more correct interpretation of the terms.

Refining Protocol Interpretations

My analysis of the core protocol structures continues. I'm now delving into the implications of the simultaneous message model. My interpretation of the provided equation is becoming clearer. I'm considering how this relates to known results, in particular the implications of partitioning rows and columns in the communication matrix. Looking for a connection to the simultaneous communication complexities.

Clarifying Simultaneous Protocols

The core of the issue now lies in properly interpreting simultaneous communication. I have determined that I must precisely define what $D^{\uparrow}(f)$ and $D^{\uparrow,\text{Bob}}(f)$ mean. From the context of the problem, I

hypothesize these are related to the number of distinct row and column vectors within the communication matrix. I am looking into how this relates to known simultaneous complexity bounds.

Reinterpreting the Variables

I am now focusing on the notation. I have assumed that $D^{\uparrow}(f)$ and $D^{\uparrow,\text{Bob}}(f)$ represent one-way communication from Alice and Bob respectively, to a referee. My hypothesis is that the simultaneous complexity is the sum of these one-way complexities, which I now believe is related to the number of distinct row and column vectors. I am working to solidify the connections.

Defining Simultaneous Communication

I am now focused on solidifying the connections between simultaneous communication protocols and the specified one-way notations. I'm exploring the implications of $D^{\uparrow}(f)$ and $D^{\uparrow,\text{Bob}}(f)$ and their relationship to the number of distinct rows and columns in the communication matrix. I intend to build a solid proof strategy for a formal proof.

Establishing Proof Strategies

I'm now formulating a plan for the formal proof. I'm focusing on defining the simultaneous protocol's essential components, followed by deriving both lower and upper bounds. My goal is to precisely connect these bounds to the provided notations. I'm building a robust and logically sound argument.

Reiterating Proof Structure

I am now focusing on the final proof. I have a clear understanding of the theorem's notation and the simultaneous communication model. I am aiming to finalize the definitions and formally present the simultaneous protocol model, with rigorous lower and upper bounds. I will make sure the proof is structured and easy to follow.

Revising Proof Framework

I am now focusing on restructuring the lower bound proof for the simultaneous communication. I am working to demonstrate the connection between the simultaneous complexity and the number of distinct rows and columns within the communication matrix. I will use a concrete example to solidify the proof's logic.

Formalizing Proof Framework

I'm now generating a well-structured LaTeX proof. My focus is on laying out definitions for all of the key terms, and clearly establishing the lower and upper bounds for the simultaneous communication complexity in terms of the number of distinct rows and columns, with specific attention to the one-way communication notation. I am prioritizing a clear, concise style.

</think>

Proof. Let M_f be the communication matrix of the function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, where the rows are indexed by inputs $x \in \{0,1\}^n$ and the columns are indexed by inputs $y \in \{0,1\}^n$. The entry (x, y) of M_f is $f(x, y)$. Let $\text{rows}(M_f)$ denote the set of distinct row vectors of M_f, and let $\text{cols}(M_f)$ denote the set of distinct column vectors of M_f.

First, we analyze the deterministic one-way communication complexity $D^{\uparrow}(f)$ (Alice to Bob). In a one-way protocol, Alice sends a message $m(x)$ to Bob, and Bob must compute $f(x, y)$. If Alice sends the same message for two inputs x and x' such that the row vectors $M_f(x, \cdot)$ and $M_f(x', \cdot)$ are distinct, there exists a y such that $f(x, y) \neq f(x', y)$. Since Bob receives the same message $m(x) = m(x')$ and possesses the same input y, he must output the same value, resulting in an error for at least one of the inputs. Therefore, Alice must send distinct messages for every distinct row vector. Conversely, if Alice sends the index of the row vector $M_f(x, \cdot)$ in the set $\text{rows}(M_f)$, Bob can reconstruct the row and determine $f(x, y)$. Thus, the optimal cost is

$$D^{\uparrow}(f) = \lceil \log_2 |\text{rows}(M_f)| \rceil.$$

By a symmetric argument, for the one-way communication complexity from Bob to Alice, denoted $D^{\uparrow,\text{Bob}}(f)$, Bob must distinguish all distinct column vectors. Thus,

$$D^{\uparrow,\text{Bob}}(f) = \lceil \log_2 |\text{cols}(M_f)| \rceil.$$

Next, we analyze the deterministic simultaneous communication complexity $D^{\parallel}(f)$. In a simultaneous protocol, Alice sends a message $m_A(x)$ and Bob sends a message $m_B(y)$ to a Referee, who computes $f(x, y)$ based solely on m_A and m_B. Let the cost be $C_A + C_B$, where $m_A \in \{0,1\}^{C_A}$ and $m_B \in \{0,1\}^{C_B}$.

Consider Alice's message function m_A. If $m_A(x) = m_A(x')$ for some x, x', then for any y and any message $m_B(y)$, the Referee receives the same pair of messages $(m_A(x), m_B(y))$ and $(m_A(x'), m_B(y))$. Consequently, the Referee outputs the same value, implying $f(x, y) = f(x', y)$ for all y. Therefore, the row vectors corresponding to x and x' must be identical. It follows that Alice must assign distinct messages to inputs having distinct row vectors. The minimum length of Alice's message is $\lceil \log_2 |\text{rows}(M_f)| \rceil$.

Similarly, if Bob sends $m_B(y) = m_B(y')$, then for any x, $f(x, y) = f(x, y')$. Thus, Bob must assign distinct messages to inputs having distinct column vectors. The minimum length of Bob's message is $\lceil \log_2 |\text{cols}(M_f)| \rceil$.

The total simultaneous complexity is the sum of the minimal message lengths required for Alice and Bob. An optimal protocol exists where Alice sends the index of the row type and Bob sends the index of the column type. The Referee, knowing the matrix structure, can determine the intersection of the row and column to output $f(x, y)$. Thus,

$$D^{\parallel}(f) = \lceil \log_2 |\text{rows}(M_f)| \rceil + \lceil \log_2 |\text{cols}(M_f)| \rceil.$$

Substituting the expressions for the one-way complexities derived above, we obtain

$$D^{\parallel}(f) = D^{\uparrow}(f) + D^{\uparrow,\text{Bob}}(f).$$

□

F.14.5 Grok

Proof. We prove the two inequalities separately.

First, we show $D^{\parallel}(f) \geq D^{\uparrow}(f) + D^{\uparrow,\text{Bob}}(f)$.

Consider an optimal deterministic simultaneous protocol Π for f with cost $D^{\parallel}(f)$, where Alice sends a message of length a depending on her input x and Bob sends a message of length b depending on his input y, with $a + b = D^{\parallel}(f)$, and a referee computes $f(x, y)$ from the pair of messages.

To obtain a one-way protocol from Alice to Bob, Alice sends the same a-bit message as in Π based on x. Bob, upon receiving this message and knowing the b-bit message he would have sent in Π based on y, can feed both messages to the referee's computation rule to obtain $f(x, y)$. This is a correct one-way protocol from Alice to Bob with cost a. Thus, $D^{\uparrow}(f) \leq a$.

Symmetrically, Bob can send his b-bit message from Π to Alice, who then uses her would-be message in Π and the received message to simulate the referee and compute $f(x, y)$. Thus, $D^{\uparrow,\text{Bob}}(f) \leq b$.

Therefore, $D^{\uparrow}(f) + D^{\uparrow,\text{Bob}}(f) \leq a + b = D^{\parallel}(f)$.

Next, we show $D^{\parallel}(f) \leq D^{\uparrow}(f) + D^{\uparrow,\text{Bob}}(f)$.

Let Π_{AB} be an optimal deterministic one-way protocol from Alice to Bob with cost $c = D^{\uparrow}(f)$, where Alice computes a c-bit message $m_{AB}(x)$ and Bob computes $f(x, y)$ from $m_{AB}(x)$ and y using a function g_{AB}.

Let Π_{BA} be an optimal deterministic one-way protocol from Bob to Alice with cost $d = D^{\uparrow,\text{Bob}}(f)$, where Bob computes a d-bit message $m_{BA}(y)$ and Alice computes $f(x, y)$ from $m_{BA}(y)$ and x using a function g_{BA}.

Fix a lexicographic ordering on $\{0,1\}^n$.

Construct a simultaneous protocol as follows: Alice sends $m_{AB}(x)$ (c bits), and Bob sends $m_{BA}(y)$ (d bits). The total cost is $c + d$.

The referee, receiving messages $m_A = m_{AB}(x)$ and $m_B = m_{BA}(y)$, proceeds as follows. Let $T = \{x' \in \{0,1\}^n \mid m_{AB}(x') = m_A\}$ (non-empty, as it contains the actual x). In Π_{AB}, all inputs in T induce identical rows in the communication matrix, since they send the same message and Bob outputs the same value for any fixed y.

The referee finds the lexicographically smallest $x' \in T$ (possible since computation is unbounded). Then $x' \in T$, so the row corresponding to x' equals the row corresponding to the actual x.

The referee computes $g_{BA}(m_B, x')$, which equals $f(x', y)$. Since the rows for x and x' are identical, $f(x', y) = f(x, y)$.

Thus, the referee correctly outputs $f(x, y)$. The total communication is $c + d = D^{\uparrow}(f) + D^{\uparrow,\mathrm{Bob}}(f)$.

Combining both directions yields the equality. □

F.15 Problem 15

F.15.1 Problem Statement

Theorem F.15. *Let f be a function and let $R^{\mathrm{ll}}(f)$ denote its randomized communication complexity and $R^{\mathrm{ll,pub}}(f)$ denote its public-coin randomized communication complexity. Then*

$$R^{\mathrm{ll}}(f) = O\left(\sqrt{n} \cdot R^{\mathrm{ll,pub}}(f)\right).$$

F.15.2 GPT

Proof. Let f be a function, and denote its randomized communication complexity by $R^{\mathrm{ll}}(f)$ and its public-coin randomized communication complexity by $R^{\mathrm{ll,pub}}(f)$. We aim to show that

$$R^{\mathrm{ll}}(f) = O\left(\sqrt{n} \cdot R^{\mathrm{ll,pub}}(f)\right).$$

First, recall the definition of randomized communication complexity. The randomized communication complexity of a function f is the minimum expected number of bits exchanged between two parties, Alice and Bob, such that they can compute $f(x, y)$, where x and y are the inputs of Alice and Bob, respectively. In this model, the communication involves both random bits and the inputs x and y.

Now, consider the case of the public-coin model, where the communication involves random bits that are shared between Alice and Bob. The public-coin randomized communication complexity, $R^{\mathrm{ll,pub}}(f)$, is defined as the minimum expected number of bits exchanged in this model. In particular, public-coin protocols can be seen as a special case of general randomized protocols where the random bits are public.

To prove the desired bound, we will use the following general facts:

1. In the standard randomized communication model, Alice and Bob can simulate any public-coin protocol by using private random bits. Specifically, if a public-coin protocol has complexity $R^{\mathrm{ll,pub}}(f)$, then a private-coin protocol may require at most $O(\sqrt{n})$ times as many bits to simulate the same protocol, where n is the size of the inputs.

2. The main idea behind the $O(\sqrt{n})$ factor is that a public-coin protocol is essentially a strategy where the randomness is shared and publicly known, whereas in a general randomized protocol, the randomness is private and not shared. The extra factor arises from the potential overhead of simulating shared randomness using private randomness.

To formalize this, we first note that the randomized communication complexity $R^{\mathrm{ll}}(f)$ can be written as the complexity of a private-coin protocol that simulates a public-coin protocol. In particular, the simulation involves a step where Alice and Bob must exchange some additional information to simulate the shared randomness. This simulation introduces an overhead factor that is at most $O(\sqrt{n})$.

Thus, we have

$$R^{\mathrm{ll}}(f) \leq O(\sqrt{n}) \cdot R^{\mathrm{ll,pub}}(f),$$

which completes the proof.

□

F.15.3 Claude

Proof. We prove that any private-coin randomized protocol can be simulated by a public-coin protocol with only a moderate increase in communication complexity.

Let π be a private-coin randomized protocol for f with communication complexity $R^{\mathrm{ll}}(f)$. In protocol π, Alice uses private random string r_A and Bob uses private random string r_B, each of length at most $\ell = R^{\mathrm{ll}}(f)$.

We construct a public-coin protocol π' that simulates π as follows. The key observation is that we can use Newman's theorem, which states that private randomness can be replaced by public randomness with an additive overhead in communication.

For any input (x, y), the protocol π computes correctly with probability at least $2/3$ over the choice of random strings (r_A, r_B). The total number of possible random string pairs is at most $2^{2\ell}$.

By a standard sampling argument, if we sample $O(\ell)$ public random strings and use them to generate candidate private random strings for both players, then with high probability, at least one of these samples will be a "good" pair that leads to a correct computation.

More precisely, we apply the following construction: Alice and Bob share a public random string which they use to generate $k = O(\ell)$ candidate pairs $(r_A^{(1)}, r_B^{(1)}), \ldots, (r_A^{(k)}, r_B^{(k)})$ of private random strings. They run the protocol π on each of these k pairs and take the majority vote of the outputs.

Since each pair gives the correct answer with probability at least $2/3$, and we have $k = O(\ell)$ independent trials, by the Chernoff bound, the majority vote gives the correct answer with probability at least $2/3$.

The communication complexity of this public-coin protocol is $k \cdot R^{\mathrm{ll}}(f) = O(\ell) \cdot R^{\mathrm{ll}}(f) = O((R^{\mathrm{ll}}(f))^2)$.

However, this gives a quadratic overhead. To obtain the stated $O(\sqrt{n})$ factor, we use a more refined analysis based on Newman's theorem with optimal parameters.

By Newman's theorem, for any $\varepsilon > 0$, we can simulate a private-coin protocol with error $1/3$ by a public-coin protocol with error $1/3 + \varepsilon$ using only $O(\log(1/\varepsilon))$ additional bits of communication.

Setting $\varepsilon = 1/3$ (which still allows for a constant error probability), we obtain that

$$R^{\mathrm{ll,pub}}(f) \leq R^{\mathrm{ll}}(f) + O(\log n).$$

For the reverse direction, we use the fact that public randomness is at least as powerful as private randomness, so $R^{\mathrm{ll,pub}}(f) \leq R^{\mathrm{ll}}(f)$.

To establish the stated relationship $R^{\mathrm{ll}}(f) = O(\sqrt{n} \cdot R^{\mathrm{ll,pub}}(f))$, we note that by Newman's theorem with optimal derandomization, the overhead of converting from public to private coins is at most $O(\log n)$ bits. When $R^{\mathrm{ll,pub}}(f) = \Omega(\sqrt{n})$, this gives the desired bound. For smaller values of $R^{\mathrm{ll,pub}}(f)$, the trivial bound $R^{\mathrm{ll}}(f) \leq n$ combined with the assumption that $R^{\mathrm{ll,pub}}(f) \geq 1$ yields $R^{\mathrm{ll}}(f) \leq n = O(\sqrt{n} \cdot \sqrt{n}) = O(\sqrt{n} \cdot R^{\mathrm{ll,pub}}(f))$ when $R^{\mathrm{ll,pub}}(f) = \Omega(\sqrt{n})$.

Therefore, we conclude that $R^{\mathrm{ll}}(f) = O(\sqrt{n} \cdot R^{\mathrm{ll,pub}}(f))$. □

F.15.4 Gemini

<think>**Defining the Problem's Scope**

I'm currently focused on defining the precise parameters of the problem. Specifically, I'm analyzing the constraints imposed by the theorem's statement. I'm carefully reviewing the definitions of $R^{\mathrm{ll}}(f)$ and all related notation to build a solid foundation. This meticulous approach should help me build a correct proof.

Revisiting the Core Premise

I'm revisiting the core statement now. The $\sqrt{n}$ factor strongly suggests a quantum connection or potentially a result specific to certain communication protocols. I need to differentiate this from Newman's Theorem. My next step is to carefully examine the provided notation, focusing on the meaning of R^{ll} to determine if it is a typo, or a niche communication model.

Interpreting the Notation

I've been meticulously examining the notation, particularly R^{ll}. Based on the $\sqrt{n}$ scaling, my initial thought was a potential connection to quantum communication or a more specific problem context, maybe a form of Grover's search. The literature suggests R^{ll} may mean "limited length" in some contexts. Currently, I'm working to narrow down the specific source of the theorem statement and its precise meaning.

Considering Problem's Origin

I'm now hunting for the source of this relationship, focusing on the notation R^{ll}. I'm exploring potential connections to discrepancy theory and quantum communication, given the $\sqrt{n}$ scaling, or a low-error

context. I'm starting to think this relates to the Disjointness problem or a related result. I will begin by searching for research on "low-error" randomized communication.

Examining Potential Origins

I'm now searching for the source material, given the theorem's focus on randomized communication. I am looking for a possible result that links private-coin and public-coin complexity with a $\sqrt{n}$ factor, possibly related to quantum communication or the discrepancy method. I've considered that the $\sqrt{n}$ factor may stem from an application of Newman's theorem or a result in low-error communication complexity.

Evaluating Newman's Theorem

I'm now back to the standard interpretation of the theorem, focusing on the relationship to Newman's theorem. I'm considering cases where public-coin complexity is bounded or large to understand if the $\sqrt{n}$ factor is reasonable. I'm examining how $\log n$ compares to $\sqrt{n}$ in this context.

Reassessing the Approach

I'm now changing my strategy to prove the statement, focusing on how Newman's Theorem can be adapted to get the suggested bound. Given that R^{pub} can be 0 or small, I'm examining how to modify the proof of Newman's theorem or any potential prior results to derive a $O(\sqrt{n})$ relationship.

Considering Newman's Application

I'm now zeroing in on a proof based on Newman's Theorem. The inequality $R(f) \leq \sqrt{n} \cdot R^{pub}(f)$ can be derived, assuming $R^{pub}(f)$ is generally greater than or equal to 1, as the original theorem is more restrictive. I plan on proving this by working from the standard definitions and Newman's Theorem. I'm now exploring edge cases.

Confirming Proof Strategy

I'm now satisfied with the proof outline, which leans heavily on a well-known result. My focus is on formally applying Newman's Theorem to the scenario, confirming all necessary conditions are met to make it work. I'll need to define the relevant communication models and establish the context before presenting the core argument.

Formalizing Proof Approach

I'm now satisfied with the proof outline, which uses Newman's Theorem. My plan is to start with a precise problem definition and assumptions, including standard models of communication complexity. Then, I'll leverage Newman's Theorem to make the connection to the stated relationship, and discuss edge cases and implications.

Refining Proof Structure

I'm now revising my proof outline, focusing on clarity. I plan to define the context, state Newman's Theorem formally, and then demonstrate how the stated inequality follows, focusing on the relationship between $\log n$ and $\sqrt{n}$ for large n. I am also carefully considering edge cases for the function.

Formulating a Concise Proof

I am now focusing on the formal structure of the proof. I will use Newman's Theorem as the basis for the argument. The challenge lies in showing the bound holds without further information about the function's complexity. The proof should address any potential edge cases related to $R^{\mathrm{ll,pub}}$.

Proof Approach Finalization

I'm now satisfied with the proof outline, which uses Newman's Theorem. My plan is to start with a precise definition of the context. I'll then formally state Newman's Theorem. Finally, I'll demonstrate how the inequality follows, and examine any edge cases.

Formalizing the Newman Argument

I am now focusing on the formal aspects of the proof, drawing from Newman's Theorem. The core argument leverages the fact that the additive $\log n$ term in Newman's bound becomes a smaller factor than the multiplicative $\sqrt{n}$ for large values of n. I am checking all edge cases to avoid making any false assumptions.

Proof Verification and Structure

I'm now satisfied with the proof outline, which uses Newman's Theorem. It's a very loose bound but it matches the prompt's condition. My immediate next step is writing out the formal proof in detail, focusing on clarifying the definitions and ensuring the logic flow. I will take special care with the edge cases to avoid making any false assumptions.

Reiterating Proof Approach

I'm now satisfied with the proof outline, which uses Newman's Theorem. It's a very loose bound but it matches the prompt's condition. My next step is writing out the formal proof in detail, emphasizing the definitions and ensuring the logic flow is clear and precise. I will give special attention to the edge cases to avoid making any false assumptions.

Confirming the Derivation

I've reviewed the proof strategy, focusing on its correctness. My approach is based on Newman's Theorem, where the $\sqrt{n}$ factor arises because $\log n$ is less than $\sqrt{n}$. The core will be a formal definition, the application of Newman's, and finally, the derivation of the bound. I will check for all possible errors.

Formulating Proof Conclusion

I'm now fully prepared to submit the proof. The logic is clear and follows from Newman's Theorem. My final adjustments are to the formatting to ensure readability and precision in the presentation, and to verify all possible edge cases.

Proving with Newman's Theorem

I'm now fully immersed in constructing the formal proof using Newman's Theorem as the basis. My key steps are defining communication models, invoking Newman's, and then showing how the requested relationship can be established, noting the additive to multiplicative gap. The edge cases are carefully checked to make sure all assumptions are met.

</think>

Proof. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function (or a partial function defined on a subset of inputs). We denote the randomized communication complexity with private coins by $R^{\mathrm{ll}}(f)$ and the randomized communication complexity with public coins by $R^{\mathrm{ll,pub}}(f)$. Standard definitions assume a bounded error probability, typically $\epsilon = 1/3$.

The relationship between public-coin and private-coin communication complexity is established by Newman's Theorem (1991). The theorem states that any public-coin protocol can be simulated by a private-coin protocol with a small additive increase in communication cost.

Let Π be an optimal public-coin protocol for f with error probability $\epsilon \leq 1/3$ and communication cost $C = R^{\mathrm{ll,pub}}(f)$. The protocol Π utilizes a shared random string r. By Newman's Theorem, there exists a multiset of random strings $S = \{r_1, r_2, \ldots, r_t\}$ such that selecting a string r_i uniformly at random from S approximates the error probability of the original protocol for every input (x, y).

Specifically, to maintain the error probability bounded by $\epsilon' < 1/2$ for all 2^{2n} possible inputs, it suffices to choose the size of the set t such that $t = O(n)$. The private-coin simulation proceeds as follows: 1. Alice chooses an index $i \in \{1, \ldots, t\}$ uniformly at random. 2. Alice sends the index i to Bob. This requires $\lceil \log_2 t \rceil$ bits. 3. Alice and Bob execute the protocol Π using r_i as the shared randomness.

The total communication cost of this private-coin protocol, denoted by Π', is:

$$\mathrm{cost}(\Pi') = \mathrm{cost}(\Pi) + \lceil \log_2 t \rceil = R^{\mathrm{ll,pub}}(f) + O(\log n).$$

Thus, we have the inequality:

$$R^{\mathrm{ll}}(f) \leq R^{\mathrm{ll,pub}}(f) + O(\log n).$$

To prove the statement $R^{\mathrm{ll}}(f) = O\left(\sqrt{n} \cdot R^{\mathrm{ll,pub}}(f)\right)$, we analyze the relationship between the additive term $O(\log n)$ and the multiplicative factor $\sqrt{n}$.

For any non-trivial function f that depends on its inputs, the communication complexity is at least constant, i.e., $R^{\mathrm{ll,pub}}(f) \geq 1$. We observe that for sufficiently large n, $\log n \leq \sqrt{n}$. Consequently, we can bound the private-coin complexity as follows:

$$R^{\mathrm{ll}}(f) \leq R^{\mathrm{ll,pub}}(f) + O(\log n) \leq R^{\mathrm{ll,pub}}(f) + O(\sqrt{n}).$$

Since $R^{\mathrm{ll,pub}}(f) \geq 1$, we have $O(\sqrt{n}) \leq O(\sqrt{n} \cdot R^{\mathrm{ll,pub}}(f))$. Therefore:

$$R^{\mathrm{ll}}(f) \leq R^{\mathrm{ll,pub}}(f) + O(\sqrt{n} \cdot R^{\mathrm{ll,pub}}(f)) = O\left(\sqrt{n} \cdot R^{\mathrm{ll,pub}}(f)\right).$$

This concludes the proof. □

F.15.5 Grok

Proof. Let $c = R^{\mathrm{ll,pub}}(f)$. There exists a public-coin simultaneous message protocol Π for f with total communication cost c and error probability at most $1/3$.

The number of possible input pairs is $N \leq 2^{2n}$. By the probabilistic method (analogous to Newman's theorem, using concentration bounds such as Chernoff and union bound), there exists a set $S = \{r_1, \ldots, r_m\}$ of $m = O(n)$ random strings such that, when fixing the public randomness to any $r_i \in S$, the resulting deterministic protocol succeeds on at least a $3/4$ fraction of input pairs (x, y). Equivalently, for every fixed input pair (x, y), at least a $3/4$ fraction of the $r_i \in S$ make Π output correctly on (x, y) when used as the public randomness.

Fix such a set S with associated indices $[m] = \{1, \ldots, m\}$, agreed upon by all parties (Alice, Bob, and the referee).

In the private-coin protocol, set $k = C\sqrt{n}$ for a sufficiently large constant $C > 0$.

Alice independently samples k indices $i_1, \ldots, i_k \in [m]$ uniformly (with replacement). For each $t = 1, \ldots, k$, she computes the message $msg_{A,t}$ that she would send in Π on input x with public randomness r_{i_t}.

She sends the k pairs $(i_t, msg_{A,t})$ to the referee.

Similarly and independently, Bob samples $j_1, \ldots, j_k \in [m]$, computes $msg_{B,s}$ for each using r_{j_s} on his input y, and sends the pairs $(j_s, msg_{B,s})$.

The communication cost per player is $k \cdot (O(\log n) + O(c)) = O(\sqrt{n}\, c)$, since describing an index requires $O(\log n)$ bits and each original message requires $O(c)$ bits.

The referee identifies the multiset of common indices (pairs where some $i_t = j_s$). For each such matching pair (t, s) with $i_t = j_s = i$, the referee computes the output that Π would produce given messages $msg_{A,t}$ and $msg_{B,s}$ (which correspond to the same r_i).

This yields a collection of outputs, one for each matching pair.

The referee outputs the majority value among these outputs (breaking ties arbitrarily if necessary).

The expected number of matchings $\lambda = k^2/m = \Theta(1)$ (by choice of C). By concentration (birthday paradox approximation and Chernoff bounds on the number of matchings and the fraction of good ones among them), for sufficiently large C, the probability that the majority is incorrect is at most $1/3$.

Thus, the protocol computes f with error at most $1/3$ and communication cost $O(\sqrt{n} \cdot c) = O(\sqrt{n} \cdot R^{\mathrm{ll,pub}}(f))$.

□

F.16 Problem 16

F.16.1 Problem Statement

Theorem F.16. $D^{\parallel}(t) = O(R^{\parallel}(t)^2)$.

F.16.2 GPT

Proof. We are given the statement $D^{\parallel}(t) = O(R^{\parallel}(t)^2)$, and we aim to prove it rigorously.

By the definition of Big-O notation, we need to show that there exist constants $C > 0$ and $t_0 \geq 0$ such that for all $t \geq t_0$,

$$|D^{\parallel}(t)| \leq C \cdot |R^{\parallel}(t)^2| \quad \text{for all} \quad t \geq t_0.$$

This is equivalent to showing that the growth of $D^{\parallel}(t)$ is bounded by a constant multiple of $R^{\parallel}(t)^2$ for sufficiently large t.

1. Assume that for sufficiently large t, there exists a constant C_1 such that

$$|D^{\parallel}(t)| \leq C_1 \cdot |R^{\parallel}(t)|^2.$$

2. To formally prove this, observe that the assumption implies there is a limit on the rate of growth of $D^{\parallel}(t)$ in terms of $R^{\parallel}(t)^2$. Specifically, for sufficiently large t, the behavior of $D^{\parallel}(t)$ is controlled by the square of $R^{\parallel}(t)$.

3. If such a constant C_1 exists, it directly follows from the definition of Big-O that $D^{\parallel}(t) = O(R^{\parallel}(t)^2)$.

Thus, we have established the required relationship between $D^{\parallel}(t)$ and $R^{\parallel}(t)^2$, and the proof is complete.

□

F.16.3 Claude

Proof. We establish the bound $D^{\parallel}(t) = O(R^{\parallel}(t)^2)$ by analyzing the relationship between the parallel displacement and parallel residual.

Let $D^{\parallel}(t)$ denote the parallel displacement at time t and $R^{\parallel}(t)$ denote the parallel residual at time t. By the fundamental relationship between displacement and residual in the parallel direction, we have

$$D^{\parallel}(t) = \int_0^t R^{\parallel}(s)\, ds.$$

Applying the Cauchy-Schwarz inequality, we obtain

$$D^{\parallel}(t) \leq \int_0^t |R^{\parallel}(s)|\, ds \leq \sqrt{t} \left(\int_0^t |R^{\parallel}(s)|^2\, ds \right)^{1/2}.$$

By the energy estimate for the parallel residual, there exists a constant $C > 0$ such that

$$\int_0^t |R^{\parallel}(s)|^2\, ds \leq C \sup_{s\in[0,t]} |R^{\parallel}(s)|^2 \cdot t.$$

Therefore,

$$D^{\parallel}(t) \leq \sqrt{t} \cdot \sqrt{Ct} \sup_{s\in[0,t]} |R^{\parallel}(s)| = \sqrt{C}\, t \sup_{s\in[0,t]} |R^{\parallel}(s)|.$$

Since the parallel residual satisfies the differential inequality $\frac{d}{dt} R^{\parallel}(t) \leq K R^{\parallel}(t)$ for some constant $K > 0$, we have by Grönwall's inequality that

$$R^{\parallel}(t) \leq R^{\parallel}(0) e^{Kt}.$$

Combining these estimates and using the fact that $t \leq \frac{1}{K}(e^{Kt} - 1) \leq \frac{e^{Kt}}{K}$ for $t \geq 0$, we obtain

$$D^{\parallel}(t) \leq \sqrt{C}\, t\, R^{\parallel}(t) \leq \frac{\sqrt{C}}{K} e^{Kt} R^{\parallel}(t).$$

By the monotonicity of $R^{\parallel}(t)$ and the bound $e^{Kt} \leq 1 + Kt + \frac{(Kt)^2}{2} e^{Kt}$, we have for sufficiently large $R^{\parallel}(t)$ that

$$D^{\parallel}(t) \leq C_1 R^{\parallel}(t)^2$$

for some constant $C_1 > 0$ depending only on C and K.

Thus, $D^{\parallel}(t) = O(R^{\parallel}(t)^2)$. □

F.16.4 Gemini

<think>**Examining the Core Problem**

I've been meticulously dissecting the formal statement, $D^{\parallel}(t) = O(R^{\parallel}(t)^2)$, breaking it down to its fundamental components. My focus is on identifying the precise mathematical relationships at play. The goal is now to discern the best approach to construct a rigorous and logically sound proof.

Deciphering Notation Context

I'm now zeroing in on the notation. My latest assessment is that the context points towards Riemannian geometry or potentially numerical analysis on manifolds. Without definitions, a direct proof is impossible. Therefore, I must infer the standard definitions associated with $D^{\parallel}(t)$ and $R^{\parallel}(t)$. My current hypothesis focuses on error analysis in parallel transport or comparing the geodesic deviation.

Revising Contextual Assumptions

I'm correcting my initial assumptions on the context. Instead of assuming the context blindly, I'm now testing it with known results. After reviewing relevant literature, I think the statement likely appears within the context of analyzing the error in parallel transport or comparing the geodesic deviation with specific numerical integration schemes on manifolds, such as "Riemannian cubic polynomials".

Synthesizing Definitive Definitions

Now, I'm defining the terms explicitly, which is crucial for a watertight argument. I'm focusing on crafting precise definitions for $D^{\|}(t)$ and $R^{\|}(t)$. Based on my research, they represent the parallel components of some deviation or error metric and the curvature, respectively. I'm aiming for an approach that links these to either parallel transport error or geodesic deviation, solidifying the proof's foundation.

Developing a Proof Framework

I'm now formulating a plan for the formal proof. I'll construct the proof utilizing Jacobi fields and incorporate the Jacobi equation. I aim to leverage Taylor series expansions around a point on a geodesic. I will then utilize the properties of the curvature tensor to establish the $O(R^{\|}(t)^2)$ bound.

Developing the Proof

I am now focusing on constructing the actual proof. I'll leverage the Jacobi equation, which relates the second covariant derivative of a Jacobi field to the curvature tensor. By carefully projecting and analyzing the resulting expressions, I'll aim to show that the tangential component of the deviation is indeed bounded by the square of a curvature-related term, thus establishing the desired asymptotic behavior.

Constructing a Formal Proof

I am now focusing on the construction of a complete and formal proof. Based on the analysis of context and notation, I will begin by carefully defining key concepts like Jacobi fields, covariant derivatives, and the curvature tensor, followed by demonstrating how the bound $D^{\|}(t) = O(R^{\|}(t)^2)$ arises from the geometrical and analytical properties of these objects, particularly around the Taylor expansion of curves, and error analysis.

Developing the Proof

I am now focusing on constructing the actual proof. I'll leverage the Jacobi equation, which relates the second covariant derivative of a Jacobi field to the curvature tensor. By carefully projecting and analyzing the resulting expressions, I'll aim to show that the tangential component of the deviation is indeed bounded by the square of a curvature-related term, thus establishing the desired asymptotic behavior.

Formulating the Core Concepts

I'm now consolidating the key concepts and constructing definitions for terms like $D^{\|}(t)$, representing the parallel component of a deviation, possibly an error metric, and $R^{\|}(t)$ the parallel component of curvature, likely related to the Riemann tensor. The framework centers around Jacobi fields and their relation to geodesic deviation or a related error. I will need to clarify the relationship between these quantities.

Synthesizing Proof Components

I'm now consolidating the key concepts and constructing definitions for terms like $D^{\|}(t)$, representing the parallel component of a deviation, possibly an error metric, and $R^{\|}(t)$, the parallel component of curvature, likely related to the Riemann tensor. I am focused on leveraging Taylor expansions in normal coordinates, and analyzing the resulting bounds, to construct the formal proof.

Formulating a Concise Strategy

I'm now consolidating the key concepts and constructing definitions for terms like $D^{\|}(t)$, representing the parallel component of a deviation, possibly an error metric, and $R^{\|}(t)$, the parallel component of curvature. Focusing on the Jacobi fields or parallel transport with the relation to geodesic deviation will be key in deriving this bound.

Revisiting Proof Construction

I'm now revisiting the construction of the formal proof, focusing on the interpretation of $R^{\|}(t)$. I'm considering it's a measure of curvature *in* the parallel direction, as opposed to the usual curvature vector. The proof will leverage Jacobi fields and Taylor expansions in normal coordinates to demonstrate the specified bound.

Revising Proof Structure

I am now focusing on the specific structure of the proof's components. Since the parallel curvature projection often vanishes, I'm examining higher-order terms in the Taylor expansion or potentially the impact of the curvature tensor's symmetries. This suggests a careful analysis is required to determine the relationship, which will be the heart of my proof.

Refining the Formal Proof

I am now focusing on refining the proof structure. I intend to formally define $P^{\|}$ as the parallel projection operator. I'll define $K(V)$ and leverage the properties of the Riemann tensor to show that $D^{\|}(t)$, being driven by the curvature, has a leading-order behavior of $O(R^{\|}(t)^2)$.

Formulating a Concise Strategy

I've been working on defining $D^{\|}(t)$ as the error or a deviation, potentially of geodesic deviation or error from integration schemes. $R^{\|}(t)$ likely relates to curvature. The proof framework will center around Jacobi fields or parallel transport with the relation to geodesic deviation to derive this bound.

Analyzing the Core Result

I'm now formulating a precise proof structure. My approach centers on the vanishing of the parallel curvature component. The aim is to demonstrate that, as a result of Riemann tensor symmetries, the leading-order contribution is zero. Therefore, I will explore the higher-order terms to relate $D^{\|}(t)$ to $R^{\|}(t)^2$.

Formulating the Proof Structure

I've been working on defining $D^{\|}(t)$ as the error or a deviation, potentially of geodesic deviation or error from integration schemes, and $R^{\|}(t)$ the parallel component of curvature, likely related to the Riemann tensor. I'm focusing on the precise definitions of these terms and the structure to relate them in a formal proof.

Formulating the Proof Structure

I am now focusing on the definition of $R^{\|}(t)$ as a measure of curvature *in* the direction parallel to the curve's tangent, and defining $D^{\|}(t)$ as a term representing the parallel component of the deviation. The proof will focus on relating the parallel component of the Jacobi equation to the symmetries of the Riemann tensor.

Re-evaluating the Proof

Now, I'm carefully analyzing the proof's core arguments, particularly focusing on the implications of the Riemann tensor symmetries. It seems critical to demonstrate that the leading-order term associated with curvature vanishes, leading to the $O(R^2)$ behavior. The next step is clearly to formalize this, and link $D^{\|}(t)$ with $R^{\|}(t)$.

Formulating the Proof Structure

I am now focusing on the definition of $R^{\|}(t)$ as a measure of curvature *in* the direction parallel to the curve's tangent, and defining $D^{\|}(t)$ as a term representing the parallel component of the deviation. The proof will focus on the relationship of the Jacobi equation to the symmetries of the Riemann tensor.

Re-evaluating the Proof

I'm now carefully analyzing the proof's core arguments, particularly focusing on the implications of the Riemann tensor symmetries. It seems critical to demonstrate that the leading-order term associated with curvature vanishes, leading to the $O(R^2)$ behavior. The next step is clearly to formalize this, and link $D^{\|}(t)$ with $R^{\|}(t)$.

Assessing the Mathematical Basis

I'm now focusing on clarifying the exact nature of $R^{\|}(t)$. My current interpretation is that this relates to the curvature effects in the problem. I'm leveraging my understanding of the Riemann curvature tensor to craft the proof.

Constructing a Formal Proof

I am now constructing a complete and formal proof. Based on the analysis of context and notation, I will begin by carefully defining key concepts like Jacobi fields, covariant derivatives, and the curvature tensor, followed by demonstrating how the bound $D^{\|}(t) = O(R^{\|}(t)^2)$ arises from the geometrical and analytical properties of these objects, particularly around the Taylor expansion of curves, and error analysis.

</think>

Proof. Let M be a Riemannian manifold with Levi-Civita connection ∇ and Riemannian curvature tensor R. Let $\gamma : I \to M$ be a smooth curve parameterized by t, with tangent vector field $T(t) = \dot{\gamma}(t)$. We assume without loss of generality that the curve is parameterized by arc length, so $\|T(t)\| = 1$.

Let $V(t)$ be a vector field along γ. We analyze the decomposition of the covariant evolution of V into components tangential (parallel) and orthogonal (normal) to the curve. The parallel component of a vector X is given by the projection $X^{\parallel} = \langle X, T\rangle T$.

Consider the curvature term appearing in the second variation of energy or the Jacobi equation, defined by the operator $\mathcal{K}(V) = R(V, T)T$. The magnitude of the deviation or error $D(t)$ is governed by the differential equation involving this curvature term. We examine the parallel component of this curvature interaction, denoted by $\mathcal{K}(V)^{\parallel}$.

Using the inner product metric, the coefficient of the parallel component is:

$$\langle \mathcal{K}(V), T\rangle = \langle R(V, T)T, T\rangle.$$

We invoke the symmetry properties of the Riemann curvature tensor. Specifically, for any vector fields X, Y, Z, W, the tensor satisfies the skew-symmetry in the last two slots:

$$\langle R(X, Y)Z, W\rangle = -\langle R(X, Y)W, Z\rangle.$$

Applying this identity to our term with $X = V$, $Y = T$, $Z = T$, and $W = T$, we obtain:

$$\langle R(V, T)T, T\rangle = -\langle R(V, T)T, T\rangle.$$

This implies that $2\langle R(V, T)T, T\rangle = 0$, and consequently:

$$\langle R(V, T)T, T\rangle = 0.$$

Thus, the first-order contribution of the curvature tensor to the dynamics in the parallel direction vanishes identically.

Let $D^{\parallel}(t)$ denote the parallel component of the deviation or displacement vector. Since the linear curvature term $\mathcal{K}(V)^{\parallel}$ is zero, the evolution of $D^{\parallel}(t)$ is not driven by terms of order $O(\|R\|)$. Instead, the non-vanishing contributions to the parallel displacement must arise from higher-order terms in the Taylor expansion of the flow or the metric. These terms involve either covariant derivatives of the curvature tensor (∇R) or quadratic interactions of the curvature tensor arising from the coupling between the normal and parallel components in the expansion of the exponential map (or the Magnus expansion of the parallel transport equation).

Assuming the curvature R is bounded, and denoting the relevant curvature bound by $R^{\parallel}(t)$ (representing the norm of the curvature operator acting on the frame), the leading non-zero term in the expansion of the parallel deviation is of the second order in the curvature. Specifically, the coupling of the normal deviation $D^{\perp} = O(R^{\parallel})$ back into the parallel equation yields a term proportional to $R^{\parallel} \cdot D^{\perp} = O((R^{\parallel})^2)$.

Therefore, we conclude that the parallel component of the deviation satisfies the asymptotic bound:

$$D^{\parallel}(t) = O(R^{\parallel}(t)^2).$$

□

F.16.5 Grok

Proof. Assume the standard definition of the time-dependent parallel diffusion coefficient in anisotropic diffusion processes, where

$$D^{\parallel}(t) = \frac{(R^{\parallel}(t))^2}{2t}.$$

Here, $R^{\parallel}(t)$ represents the root mean squared parallel displacement at time t, so $(R^{\parallel}(t))^2 = \langle (x^{\parallel}(t))^2\rangle$.

We need to show that $D^{\parallel}(t) = O((R^{\parallel}(t))^2)$, meaning there exist constants $C > 0$ and $t_0 > 0$ such that for all $t \geq t_0$,

$$D^{\parallel}(t) \leq C(R^{\parallel}(t))^2.$$

Substitute the definition:

$$D^{\parallel}(t) = \frac{1}{2t}(R^{\parallel}(t))^2.$$

Choose $t_0 = 1$ (assuming t is measured in units where this is appropriate; the choice is arbitrary but fixed). For $t \geq 1$,

$$\frac{1}{2t} \leq \frac{1}{2}.$$

Thus,

$$D^{\parallel}(t) \leq \frac{1}{2}(R^{\parallel}(t))^2.$$

Setting $C = \frac{1}{2}$, the inequality holds for all $t \geq 1$. If $R^{\parallel}(t) = 0$ for some t, the inequality is trivial. Therefore, $D^{\parallel}(t) = O((R^{\parallel}(t))^2)$. □

F.17 Problem 17

F.17.1 Problem Statement

4. Determine $R^{II}(\mathsf{eq})$.

F.17.2 GPT

Proof. We are tasked with determining $R^{II}(\mathsf{eq})$. To do so, we will proceed step by step, carefully evaluating the necessary components and applying the relevant definitions and properties.

First, recall that R^{II} is defined as the second iteration of the relation R applied to the element eq. By the definition of the second iteration, we have:

$$R^{II}(\mathsf{eq}) = R(R(\mathsf{eq})).$$

Thus, to compute $R^{II}(\mathsf{eq})$, we must first determine $R(\mathsf{eq})$, and then apply the relation R again to the result.

Let $R(\mathsf{eq}) = x$ for some element x. This gives us the intermediate result:

$$R^{II}(\mathsf{eq}) = R(x).$$

Now, we proceed to find the explicit value of $x = R(\mathsf{eq})$. Depending on the specific structure of the relation R and the properties of the element eq, we analyze the result of applying R to eq.

Without loss of generality, assume that $R(\mathsf{eq})$ produces a well-defined result. By applying R again to this result, we compute $R^{II}(\mathsf{eq}) = R(x)$.

If the relation R has a specific known form, such as a binary relation on a set A, where the operation R applies to pairs $(a, b) \in A \times A$, then we may explicitly compute $R^{II}(\mathsf{eq})$ by evaluating this operation step by step, using the known properties of the relation.

Finally, having determined $R^{II}(\mathsf{eq})$, we conclude the proof. Therefore, we have:

$$R^{II}(\mathsf{eq}) = \boxed{R(x)}.$$

□

F.17.3 Claude

Proof. We determine the Ramsey number $R^{II}(\mathsf{eq})$, which is the minimum number of colors needed such that for any 2-coloring of pairs from an infinite set, there exists an infinite monochromatic subset with the equivalence relation property.

First, we establish that $R^{II}(\mathsf{eq}) \leq 2$. Consider any 2-coloring $\chi : [\omega]^2 \to \{0, 1\}$ of pairs from the natural numbers. By the infinite Ramsey theorem, there exists an infinite homogeneous set $H \subseteq \omega$ such that all pairs in $[H]^2$ receive the same color. Without loss of generality, assume all pairs in $[H]^2$ are colored 0. The coloring restricted to H trivially satisfies the equivalence relation property: reflexivity holds vacuously for pairs, symmetry is immediate since $\chi(\{x, y\}) = \chi(\{y, x\})$, and transitivity holds

since for any $x, y, z \in H$, we have $\chi(\{x, y\}) = \chi(\{y, z\}) = \chi(\{x, z\}) = 0$. Thus H forms an infinite monochromatic set with the equivalence relation structure.

Next, we show that $R^{II}(\mathsf{eq}) \geq 2$. We must demonstrate that a single color is insufficient. Consider a 1-coloring where all pairs receive color 0. While this trivially gives an infinite monochromatic set (the entire domain), the key observation is that we need at least 2 colors to force the existence of a non-trivial equivalence structure. More precisely, consider the partition of ω into two infinite sets A and B. Define a coloring by: color pairs within A with 0, pairs within B with 0, and pairs crossing between A and B with 1. Any infinite homogeneous set must be contained entirely within A or entirely within B, and such a set has all its pairs colored 0, forming a valid equivalence class. However, with only 1 color available, we cannot distinguish between different equivalence classes, which contradicts the requirement that the coloring must support the equivalence relation structure across all possible configurations.

By the infinite Ramsey theorem $R^{II}(2) = 2$, and since the equivalence relation property is preserved under the standard Ramsey argument for 2 colors, we conclude that exactly 2 colors are both necessary and sufficient.

Therefore, $R^{II}(\mathsf{eq}) = 2$. □

F.17.4 Gemini

<think>**Examining the Core Task**

I've just begun dissecting the request. My primary focus is understanding the formalized statement and the implied objective: deriving $R^{II}(\mathsf{eq})$. I'm zeroing in on the specific notation to clarify the context and the problem's scope. Further work will identify essential mathematical tools.

Pinpointing the Notation's Meaning

I am now delving into the core notation. Decoding the "II" and "eq" is crucial for framing a relevant response. I'm focusing on "II" to identify if it specifies a particular reduction strategy or construction, likely related to the proof of confluence. The "eq" seems to denote an equivalence relation or specific redexes, and understanding that is my next priority.

Deconstructing the Symbols

I'm now zeroing in on the symbols' precise meanings. I suspect this problem originates from a specific text on term rewriting or, less likely, a particular system of mathematical logic. I'm focusing on "II" to understand its operational significance, perhaps representing a specific construction within a proof, as in a reduction strategy. I'm also examining the literature on finite model theory and presburger arithmetic, as this seems relevant to understanding image relations and closures.

Refining the Contextual Scope

I'm now prioritizing the specific context. My recent searches point strongly to connections with Ebbinghaus, Flum, and Thomas's work in Mathematical Logic, and exercises related to partial isomorphisms or Ehrenfeucht-Fraïssé games. The Pol-Inv Galois connection in Post's Lattice is also a contender. My next step will focus on these areas to deduce the precise meaning.

Revising Contextual Interpretations

My initial attempts to define R^{II} as relating to the *image* of a functor or *reductions* strategies have been unsuccessful. Now I am leaning toward interpreting R^{II} as defining relations of rank two or second-order relations, as found in a second-order logic context. I'm exploring the potential for R to represent a set or family of relations, likely based on material from Monk's *Mathematical Logic* or similar.

Identifying Potential Contexts

My exploration is now shifting to the most likely domains for this problem. Given the notation, "Mathematical Logic" by Monk is a key contender. In relation algebra, R might denote a set or family of relations, where the superscript defines some property, e.g. the rank. Alternatively, in the Tarski/Givant calculus, R^{II} might be a key term.

Identifying the Key Result

I'm now zeroing in on the anticipated outcome: a specific set of binary relations. The "Determine" prompt now seems to point directly towards characterizing the invariant relations under the full symmetric

group, acting on the equality relation. I'm focusing on the standard orbit partition and the resulting classification.

Clarifying Invariant Relations

My efforts now center on interpreting the meaning of "R^{II}" within the context of the problem, with the help of "Determining $R^{II}(\mathsf{eq})$". Specifically, I'm now leaning toward the idea of a relationship between binary relations and automorphisms, with the full symmetric group acting on the equality relation. Therefore, I'll attempt to relate the result to its potential connection with orbit structure.

Identifying the Core Concept

I'm now zeroing in on the essence of the problem. My analysis points to a connection between binary relations and automorphisms, with the full symmetric group S_Ω acting on equality. This suggests the result involves a classification based on orbit structures. I anticipate four specific relations as the final result, in line with prior expectations.

Defining the Core Problem

I am now structuring the essential steps, after carefully considering the question's scope. The task is to determine the complete set of relations, specifically $R^{II}(\mathsf{eq})$, which has me focused on the algebraic structures related to the symmetry group S_Ω acting on equality, and invariant properties of said relations. I'm focusing on the general group action and associated orbit partition, expecting a concise result.

Clarifying Invariance and Orbits

I'm now focusing on the implications of invariance under the automorphism group. The core task is to identify the orbits of the symmetric group S_Ω acting on pairs of elements, and then define the resulting set of invariant binary relations. I anticipate that the partition of the domain is the key to demonstrating the result.

Revising the Anticipated Outcome

I'm now certain of the problem's core, with a high degree of confidence. The goal is now to determine the set of invariant relations under the action of the full symmetric group on the equality relation. The anticipated outcome consists of four relations.

Formulating the Proof Structure

I'm now consolidating the thought process, focusing on constructing the formal proof. I'll define the framework, specifically the automorphism group and associated orbits, to clearly determine the structure of $R^{II}(\mathsf{eq})$. I'll start with defining the action of the symmetric group on the cartesian product.

Formulating the Proof

I am now structuring the essential steps, after carefully considering the question's scope. The core of my efforts will be writing out the proof itself, precisely detailing the action of the automorphism group on the binary relations. This involves clear definitions and identification of orbits to establish the final result.

Constructing a Formal Proof

I am now focusing on the formal structure of the proof. Defining the notation, the action of the automorphism group on the binary relations, and then determining the resulting invariant relations is my next task. I am working towards an elegant structure.

</think>

Proof. Let A be a set with $|A| \geq 2$. We interpret $R^{II}(\mathsf{eq})$ as the set of binary relations on A that are invariant under the automorphisms of the equality relation $\mathsf{eq} = \{(x, x) \mid x \in A\}$. Let $G = \mathrm{Aut}(\langle A, \mathsf{eq}\rangle)$ be the group of permutations $\pi : A \to A$ that preserve eq. A permutation π preserves eq if for all $x, y \in A$, $x = y \iff \pi(x) = \pi(y)$. Since π is a bijection, this condition is trivially satisfied for all permutations. Thus, $G = S_A$, the symmetric group on A.

The set $R^{II}(\mathsf{eq})$ consists of those relations $R \subseteq A \times A$ such that for all $\pi \in S_A$, $\pi(R) = R$, where $\pi(R) = \{(\pi(x), \pi(y)) \mid (x, y) \in R\}$. Such relations are exactly the unions of the orbits of the action of S_A on $A \times A$. We determine these orbits by considering the action of S_A on pairs $(x, y) \in A \times A$.

Consider the diagonal set $\Delta = \{(x, x) \mid x \in A\}$. For any $(x, x) \in \Delta$ and any $\pi \in S_A$, $\pi \cdot (x, x) = (\pi(x), \pi(x)) \in \Delta$. Furthermore, for any two diagonal elements (x, x) and (y, y), there exists a permutation $\pi \in S_A$ such that $\pi(x) = y$ (since S_A acts transitively on A). Thus, Δ constitutes a single orbit.

Consider the off-diagonal set $\Delta^c = \{(x, y) \in A \times A \mid x \neq y\}$. For any $(x, y) \in \Delta^c$ and $\pi \in S_A$, since π is injective, $\pi(x) \neq \pi(y)$, so $\pi \cdot (x, y) \in \Delta^c$. Furthermore, for any two pairs (x, y) and (u, v) in Δ^c, we have $x \neq y$ and $u \neq v$. Since $|A| \geq 2$, the symmetric group S_A is 2-transitive, meaning there exists a permutation $\pi \in S_A$ such that $\pi(x) = u$ and $\pi(y) = v$. Thus, Δ^c constitutes a single orbit.

The invariant relations are the possible unions of these orbits. The set of orbits is $\{\Delta, \Delta^c\}$. The distinct unions are: 1. The empty set: $\emptyset$. 2. The diagonal: Δ (the relation eq). 3. The off-diagonal: Δ^c (the relation $\neq$). 4. The total set: $\Delta \cup \Delta^c = A \times A$ (the universal relation, denoted $\top$ or ∇).

Therefore, we determine that

$$R^{II}(\mathsf{eq}) = \{\emptyset, \mathsf{eq}, \neq, A \times A\}.$$

□

F.17.5 Grok

Proof. The function $\mathsf{eq} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is defined by $\mathsf{eq}(x, y) = 1$ if and only if $x = y$. The complexity $R^{II}(\mathsf{eq})$ denotes the public-coin randomized simultaneous-message communication complexity of eq, where Alice and Bob send messages simultaneously to a referee, with bounded error at most $1/3$.

We prove that $R^{II}(\mathsf{eq}) = \Theta(1)$.

For the upper bound $R^{II}(\mathsf{eq}) = O(1)$, consider the following protocol. Let $k = 3$, so $2^{-k} = 1/8 < 1/3$. Using public randomness, generate k independent uniform random vectors $r_1, \ldots, r_k \in \{0,1\}^n$ (equivalently, a random $k \times n$ matrix R over $\mathbb{F}_2$).

Alice computes the k-bit vector $a = (\langle r_j, x \rangle)_{j=1}^k$, where $\langle u, v \rangle = \sum_{i=1}^n u_i v_i \pmod 2$, and sends a to the referee.

Bob similarly computes and sends $b = (\langle r_j, y \rangle)_{j=1}^k$.

The referee outputs 1 if $a = b$ and 0 otherwise.

If $x = y$, then $a = b$ with certainty, so the output is correctly 1.

If $x \neq y$, let $v = x + y \in \mathbb{F}_2^n$ (componentwise addition modulo 2), so $v \neq 0$. Then $\langle r_j, x \rangle = \langle r_j, y \rangle$ if and only if $\langle r_j, v \rangle = 0$. For each fixed nonzero v, since r_j is uniform random, $\langle r_j, v \rangle$ is a uniform random bit in $\{0, 1\}$. Moreover, the bits for different j are independent. Thus,

$$\Pr[a = b] = \Pr[\langle r_j, v \rangle = 0 \, \forall \, j = 1, \ldots, k] = 2^{-k} = 1/8 < 1/3.$$

The protocol errs only on unequal inputs, with probability at most $1/8 < 1/3$, and is correct with probability 1 on equal inputs. Each message has length $k = 3 = O(1)$, so the total communication is $O(1)$.

For the lower bound $R^{II}(\mathsf{eq}) = \Omega(1)$, suppose for contradiction that there exists a protocol with total communication cost 0 (messages of length 0). Then the referee receives no information and must output a fixed bit, independent of the inputs. If the fixed output is 1, the protocol errs on all unequal inputs (x, y) with $x \neq y$. If the fixed output is 0, it errs on all equal inputs (x, x). In either case, there exist inputs on which the error probability is $1 > 1/3$, a contradiction. Thus, positive communication is required in the worst case, so $R^{II}(\mathsf{eq}) = \Omega(1)$.

Therefore, $R^{II}(\mathsf{eq}) = \Theta(1)$. □

F.18 Problem 18

F.18.1 Problem Statement

Theorem F.17 (Exercise 4.25)**.** *Recall the definition of t^m (Section 4.1). Prove that, for all k and m,*

$$D^k(t) \leq D^k(t^m) \leq D^k(t^m) \leq m \cdot D^k(t).$$

F.18.2 GPT

Proof. We are tasked with proving the following inequality for all k and m:

$$D^k(t) \leq D^k(t^m) \leq m \cdot D^k(t).$$

Step 1: Establishing $D^k(t) \leq D^k(t^m)$

Recall the definition of t^m (Section 4.1). Let us start by considering the derivative of t^m with respect to t. Using the chain rule for derivatives, we have:

$$D(t^m) = mt^{m-1}.$$

Now, applying the k-th derivative to t^m, we obtain:

$$D^k(t^m) = m(m-1)(m-2)\cdots(m-k+1)t^{m-k}.$$

Note that this expression is non-negative for $t \geq 0$. Therefore, for any k, we have:

$$D^k(t^m) \geq D^k(t),$$

since the derivatives of t^m grow with m and the function t^{m-k} is greater than or equal to t^0 for non-negative t. Hence, we conclude:

$$D^k(t) \leq D^k(t^m).$$

Step 2: Establishing $D^k(t^m) \leq m \cdot D^k(t)$

Now, we wish to show that:

$$D^k(t^m) \leq m \cdot D^k(t).$$

From the earlier result, we already know that:

$$D^k(t^m) = m(m-1)(m-2)\cdots(m-k+1)t^{m-k}.$$

We can factor out m from the product and observe that the remaining terms $(m-1), (m-2), \ldots, (m-k+1)$ are all less than or equal to m. Thus, we can bound the k-th derivative of t^m as:

$$D^k(t^m) \leq m^k t^{m-k}.$$

Since t^{m-k} is non-negative and t^m is increasing in t, we now note that:

$$D^k(t) = t^k.$$

Therefore:

$$D^k(t^m) \leq m \cdot D^k(t),$$

since the factor m^k in the derivative of t^m leads to the inequality $D^k(t^m) \leq m \cdot D^k(t)$.

Conclusion

Combining the results of Step 1 and Step 2, we have shown that:

$$D^k(t) \leq D^k(t^m) \leq m \cdot D^k(t).$$

Thus, the theorem is proven. □

F.18.3 Claude

Proof. We first note that there appears to be a typographical error in the statement, as the middle inequality is repeated. The intended statement should be:

$$D^k(t) \leq D^k(t^m) \leq m \cdot D^k(t).$$

Recall from Section 4.1 that t^m denotes the tree obtained by taking m copies of tree t and joining them at a common root. More precisely, if t has root r, then t^m consists of a new root r' with m children, each of which is the root of an isomorphic copy of t.

Let $D^k(t)$ denote the number of nodes at depth k in tree t.

Proof of the left inequality: $D^k(t) \leq D^k(t^m)$.

For $k = 0$, we have $D^0(t) = 1$ (the root of t) and $D^0(t^m) = 1$ (the root of t^m), so $D^0(t) = D^0(t^m)$.

For $k \geq 1$, the nodes at depth k in t^m consist of all nodes at depth $k-1$ in the m copies of t that are children of the root of t^m. Since each copy of t in t^m is rooted at depth 1 in t^m, nodes at depth k in t^m correspond to nodes at depth $k-1$ in the original tree t.

However, we need to reconsider this carefully. For $k = 1$, the nodes at depth 1 in t^m are the m roots of the copies of t, while $D^1(t)$ counts the children of the root in t.

More precisely, for $k \geq 1$, nodes at depth k in t^m are exactly the nodes at depth $k-1$ in each of the m copies of t. Therefore:

$$D^k(t^m) = m \cdot D^{k-1}(t) \text{ for } k \geq 1.$$

For $k = 0$, we have $D^0(t) = D^0(t^m) = 1$.

For $k \geq 1$, since $D^{k-1}(t) \geq 0$ and $m \geq 1$, we need to show $D^k(t) \leq m \cdot D^{k-1}(t)$.

Actually, let us reconsider the structure more carefully. The nodes at depth k in t^m for $k \geq 1$ are precisely the union of nodes at depth $k-1$ in each of the m copies of t (where depth is measured from the root of each copy). Thus $D^k(t^m) = m \cdot D^{k-1}(t)$ for $k \geq 1$.

Since t is a tree with at least one node, for $k \geq 1$, if $D^k(t) > 0$, then $D^{k-1}(t) \geq D^k(t)/b$ where b is the branching factor. In general, $D^k(t) \leq m \cdot D^{k-1}(t)$ when m is sufficiently large or when considering the tree structure.

For the inequality $D^k(t) \leq D^k(t^m) = m \cdot D^{k-1}(t)$, this holds when $k = 0$ trivially. For $k \geq 1$, since $m \geq 1$ and $D^{k-1}(t) \geq 1$ when $D^k(t) > 0$, we have $D^k(t) \leq m \cdot D^{k-1}(t)$ for sufficiently large m.

Proof of the right inequality: $D^k(t^m) \leq m \cdot D^k(t)$.

For $k = 0$, we have $D^0(t^m) = 1 \leq m \cdot 1 = m \cdot D^0(t)$ since $m \geq 1$.

For $k \geq 1$, as established above, $D^k(t^m) = m \cdot D^{k-1}(t)$. We need $m \cdot D^{k-1}(t) \leq m \cdot D^k(t)$, which requires $D^{k-1}(t) \leq D^k(t)$. This holds for trees where the number of nodes increases with depth up to depth k, which is the case for complete trees and the intended context.

Given the standard interpretation where both inequalities hold with equality in certain cases, we conclude the proof. □

F.18.4 Gemini

<think>**Initiating Analysis Protocol**

I'm focused on the theorem statement, pinpointing its core components, specifically the interplay between t^m and D^k. My initial step involves deconstructing the formal statement to fully grasp the precise relationships and potential implications. This involves scrutinizing the parameters and conditions described within the theorem.

Refining Theorem Comprehension

I'm now carefully reviewing the definitions and notation in Levin, Peres, and Wilmer, to establish a rock-solid understanding of $D^k(t)$ and t. My focus is on the specific meaning of $D^k(t)$ in the context of the book and identifying the precise conditions under which this inequality holds. The next step involves connecting the provided theorem with the existing concepts in the text.

Unveiling Notation Ambiguity

I'm now grappling with the discrepancy between the prompt's notation ($D^k(t)$ and t^m) and the standard usage in Levin, Peres, and Wilmer. The goal is to accurately identify the textbook being referenced and

determine the precise meaning of $D^k(t)$ and how it relates to t^m, as both seem non-standard. The plan is to search for exercise 4.25 and definitions related to it.

Pinpointing Theorem's Context

I'm now investigating potential textbooks and specific sections, focusing on the notation. My assessment leans toward concentration inequalities, maybe in connection to transportation cost. I'm exploring the definitions of $D^k(t)$ and t^m in different texts, like *Concentration Inequalities* by Boucheron, Lugosi, and Massart, trying to connect them with the prompt.

Clarifying Theorem Components

I'm now fully immersed in the exercise. It's clear that the prompt refers to Exercise 4.25 from the second edition of Levin, Peres, and Wilmer's *Markov Chains and Mixing Times*. I've found it! The goal is now to accurately interpret and prove the inequality, which should be relatively straightforward, based on the definitions in Section 4.1. I'll need to define τ and $\tau^{(m)}$ which should be clear from the definition.

Defining τ and $\tau^{(m)}$

I'm now zeroing in on Levin, Peres, and Wilmer (2017), Exercise 4.25, after confirming the context. The next step is to carefully interpret the notation, focusing on defining τ and $\tau^{(m)}$ as described in Section 4.1. This is the crucial foundation for demonstrating the stated inequality, which should follow from basic properties of coupling or mixing times. I'll meticulously review the relevant definitions.

Unraveling Inequality's Core

Now that the context is confirmed, I'm focusing on the meaning of $\tau^{(m)}$ within the framework of Exercise 4.25 in Levin/Peres. I'm carefully reviewing Section 4.1 in *Markov Chains and Mixing Times* for the definitions of τ and $\tau^{(m)}$, as they are central to proving the inequality. With a precise definition, I can begin to construct the proof.

Decoding Exercise 4.25

I'm now zeroing in on Levin, Peres, and Wilmer, Exercise 4.25, and specifically the definitions in Section 4.1. I'm carefully reviewing the definitions of τ, $\tau^{(m)}$, and D^k, ensuring a solid grasp for the proof. The goal is to connect the inequality's structure with the concepts of coupling and mixing times in the textbook.

Defining the Variables

I'm now firmly focusing on the correct book, Levin, Peres, and Wilmer's *Markov Chains and Mixing Times*, and the relevant exercise. I'm carefully connecting the meaning of τ and $\tau^{(m)}$ within the framework to each other. The goal is to accurately interpret and prove the inequality in the exercise.

Unveiling Exercise's Context

I'm now very confident that the prompt refers to Exercise 4.25 in Levin, Peres, and Wilmer, from the *Markov Chains and Mixing Times* text. This clarifies the relevant definitions of τ, $\tau^{(m)}$, and, most importantly, the exact inequality I must prove. Now, I'll concentrate on the specific definitions.

Reviewing Exercise 4.25

I'm now thoroughly examining Exercise 4.25 from Levin, Peres, and Wilmer, as the context is now clear. I'm carefully reviewing the definitions of the involved variables, particularly τ and $\tau^{(m)}$, and their connections to the core inequality. My objective is to formulate a clear, concise strategy for proving it.

Defining Mixing Time Variables

I'm now fully immersed in the exercise. My focus is on Levin, Peres, and Wilmer (2017), Exercise 4.25, and understanding the variables within. Having established that, I'm now aiming to accurately define and relate the τ and $\tau^{(m)}$ notation of the textbook to formulate the proof.

Decoding the Variables

I'm now fully immersed in the exercise. I've pinpointed the relevant section in Levin/Peres. I'm focusing on accurately representing t and t^m and their relationship, ensuring a clear understanding of what $D^k(t)$ and $D^k(t^m)$ represent. I will begin by defining the mixing time and coupling time within the exercise's framework.

Proof Strategy Formation

I'm now formulating a precise proof structure. The key is accurately interpreting the notation t, t^m, and $D^k(t)$. My approach involves using the product chain of Markov chains, as it matches the inequality

structure. I'll need to clearly define these within my proof.

Finalizing the Proof Structure

I'm now satisfied with the core components and have established a clear plan. I'll utilize the product chain framework, linking $D^k(t)$ and $D^k(t^m)$ to the total variation distance. The lower bound proof will leverage the projection argument, and the upper bound will employ a coupling strategy. I am ready to begin writing the detailed proof.

Finalizing the Analysis

I'm now fully prepared to write the formal proof, armed with a clear understanding of the theorem statement and its relevant components, including the meaning of t, t^m, and $D^k(t)$. The structure is set, and the stage is set for a concise demonstration, carefully leveraging coupling and projection techniques for both the upper and lower bounds.

Proof Development Commenced

I am now structuring the proof, focusing on the inequality involving $D^k(t)$ and $D^k(t^m)$ and applying the coupling argument for the upper bound and the projection argument for the lower bound. I will define t and t^m carefully.

Refining the Proof's Core

Now that the context is clear, I'm fully constructing the formal proof, leveraging coupling and projection techniques for upper and lower bounds. My goal is to use the existing framework while maintaining absolute clarity and precision in defining the product chain's operation and the total variation distance.

Constructing Formal Proof

I am now structuring the formal proof, including precise definitions for all parameters. This will involve showing both $D^k(t) \leq D^k(t^m)$ and then $D^k(t^m) \leq mD^k(t)$. The goal is to provide a comprehensive, rigorous proof.

Constructing Formal Proof

I am now structuring the formal proof, including precise definitions for all parameters. This will involve showing both $D^k(t) \leq D^k(t^m)$ and then $D^k(t^m) \leq mD^k(t)$. The goal is to provide a comprehensive, rigorous proof.

Formulating the Final Proof

I am now thoroughly developing the formal proof. I'm focusing on defining t and t^m carefully as the Markov chain and the product chain, respectively. The strategy will leverage the coupling and projection techniques. I'm preparing to ensure the proof is detailed and self-contained, using all notations previously established.

Formalizing the Proof

I'm now fully immersed in the process of drafting the complete formal proof. I'm leveraging my understanding of the theorem's core principles, including the total variation distance, projection, and coupling techniques. The goal is to provide a concise and precise proof.

Formulating the Proof Structure

I'm now putting the finishing touches on the formal proof. I'm leveraging definitions of the TV distance, projection, and the coupling method. I aim for concision, while defining all key components to guarantee a self-contained demonstration.

Constructing Formal Proof

I'm now fully immersed in the process of drafting the complete formal proof. I'm leveraging my understanding of the theorem's core principles, including the total variation distance, projection, and coupling techniques. The goal is to provide a concise and precise proof.

Constructing the Formal Proof

I am now thoroughly developing the formal proof. I'm focusing on defining t and t^m carefully as the Markov chain and the product chain, respectively. The strategy will leverage the coupling and projection techniques. I'm preparing to ensure the proof is detailed and self-contained, using all notations previously established.

Constructing the Proof's Core

Now that the proof's foundation has been clearly established, I'm carefully writing out the detailed arguments, including the projection technique to obtain the lower bound and a robust coupling construction

for the upper bound. My objective is to create a complete and self-contained proof.
</think>

Proof. Let t represent a Markov chain on a state space Ω with transition matrix P and stationary distribution π. The quantity $D^k(t)$ denotes the maximum total variation distance from stationarity at time k, defined as:

$$D^k(t) = \max_{x \in \Omega} \|P^k(x, \cdot) - \pi\|_{\text{TV}}.$$

The notation t^m denotes the product chain consisting of m independent copies of the chain t. The state space of t^m is Ω^m, the transition matrix is $P^{\otimes m}$, and the stationary distribution is $\pi^{\otimes m}$. The distance for the product chain is given by:

$$D^k(t^m) = \max_{\mathbf{x} \in \Omega^m} \|(P^{\otimes m})^k(\mathbf{x}, \cdot) - \pi^{\otimes m}\|_{\text{TV}}.$$

Lower Bound: $D^k(t) \leq D^k(t^m)$.

Let $x \in \Omega$ be a state that maximizes the distance for the single chain, i.e., $\|P^k(x, \cdot) - \pi\|_{\text{TV}} = D^k(t)$. Consider the starting state $\mathbf{x} = (x, x, \ldots, x) \in \Omega^m$ for the product chain. Let $A \subseteq \Omega$ be an event. We define the cylindrical event $E = A \times \Omega^{m-1} \subseteq \Omega^m$. The marginal distribution of the first component of the product chain is independent of the other components and evolves according to P. Thus,

$$(P^{\otimes m})^k(\mathbf{x}, E) = P^k(x, A) \quad \text{and} \quad \pi^{\otimes m}(E) = \pi(A).$$

By the definition of the total variation distance as $\sup_S |\mu(S) - \nu(S)|$, we have:

$$\|(P^{\otimes m})^k(\mathbf{x}, \cdot) - \pi^{\otimes m}\|_{\text{TV}} \geq |(P^{\otimes m})^k(\mathbf{x}, E) - \pi^{\otimes m}(E)| = |P^k(x, A) - \pi(A)|.$$

Taking the supremum over all $A \subseteq \Omega$, we obtain:

$$\|(P^{\otimes m})^k(\mathbf{x}, \cdot) - \pi^{\otimes m}\|_{\text{TV}} \geq \|P^k(x, \cdot) - \pi\|_{\text{TV}} = D^k(t).$$

Since $D^k(t^m)$ is the maximum over all $\mathbf{x} \in \Omega^m$, it follows that $D^k(t^m) \geq D^k(t)$.

Upper Bound: $D^k(t^m) \leq m \cdot D^k(t)$.

We use the coupling characterization of the total variation distance: $\|\mu - \nu\|_{\text{TV}} = \inf\{\mathbb{P}(X \neq Y)\}$, where the infimum is taken over all couplings (X, Y) of μ and ν.

Let $\mathbf{x} = (x_1, \ldots, x_m) \in \Omega^m$ be an arbitrary starting state. For each coordinate $i \in \{1, \ldots, m\}$, let $(X_k^{(i)}, Y_k^{(i)})$ be an optimal coupling of the distributions $P^k(x_i, \cdot)$ and π such that:

$$\mathbb{P}(X_k^{(i)} \neq Y_k^{(i)}) = \|P^k(x_i, \cdot) - \pi\|_{\text{TV}}.$$

By definition, $\|P^k(x_i, \cdot) - \pi\|_{\text{TV}} \leq D^k(t)$ for all i.

We construct a coupling $(\mathbf{X}_k, \mathbf{Y}_k)$ for the product chain by taking the independent product of these component couplings. That is, $\mathbf{X}_k = (X_k^{(1)}, \ldots, X_k^{(m)})$ and $\mathbf{Y}_k = (Y_k^{(1)}, \ldots, Y_k^{(m)})$, where the pairs for different i are independent. Then $\mathbf{X}_k \sim (P^{\otimes m})^k(\mathbf{x}, \cdot)$ and $\mathbf{Y}_k \sim \pi^{\otimes m}$.

The vectors $\mathbf{X}_k$ and $\mathbf{Y}_k$ are unequal if and only if they differ in at least one coordinate:

$$\{\mathbf{X}_k \neq \mathbf{Y}_k\} = \bigcup_{i=1}^{m} \{X_k^{(i)} \neq Y_k^{(i)}\}.$$

Applying the union bound (Boole's inequality):

$$\mathbb{P}(\mathbf{X}_k \neq \mathbf{Y}_k) \leq \sum_{i=1}^{m} \mathbb{P}(X_k^{(i)} \neq Y_k^{(i)}).$$

Substituting the bound for each component:

$$\mathbb{P}(\mathbf{X}_k \neq \mathbf{Y}_k) \leq \sum_{i=1}^{m} \|P^k(x_i, \cdot) - \pi\|_{\text{TV}} \leq \sum_{i=1}^{m} D^k(t) = m \cdot D^k(t).$$

By the coupling inequality, the total variation distance is bounded by the probability of disagreement:

$$\|(P^{\otimes m})^k(\mathbf{x}, \cdot) - \pi^{\otimes m}\|_{\text{TV}} \leq \mathbb{P}(\mathbf{X}_k \neq \mathbf{Y}_k) \leq m \cdot D^k(t).$$

Since this holds for any starting state $\mathbf{x} \in \Omega^m$, we conclude:

$$D^k(t^m) = \max_{\mathbf{x} \in \Omega^m} \|(P^{\otimes m})^k(\mathbf{x}, \cdot) - \pi^{\otimes m}\|_{\text{TV}} \leq m \cdot D^k(t).$$

Combining the bounds, we have $D^k(t) \leq D^k(t^m) \leq m \cdot D^k(t)$. □

F.18.5 Grok

Proof. The function t^m is defined on inputs consisting of m disjoint blocks, where Alice receives $(x_1, \ldots, x_m)$ and Bob receives $(y_1, \ldots, y_m)$, with each (x_i, y_i) being an input to t, and

$$t^m((x_1, \ldots, x_m), (y_1, \ldots, y_m)) = \bigwedge_{i=1}^{m} t(x_i, y_i).$$

Equivalently, the communication matrix of t^m is the m-fold Kronecker product of the communication matrix of t.

First, prove $D^k(t^m) \geq D^k(t)$.

Let Π be any deterministic protocol computing t^m using at most k rounds and communication cost $D^k(t^m)$.

To compute t, fix inputs (x_j^*, y_j^*) for $j = 2, \ldots, m$ such that $t(x_j^*, y_j^*) = 1$ (such fixed inputs exist if $t^{-1}(1)$ is nonempty; otherwise both functions are trivial and the inequality holds with equality 0).

On inputs (x, y) to t, Alice sets her input to $(x, x_2^*, \ldots, x_m^*)$ and Bob sets his to $(y, y_2^*, \ldots, y_m^*)$. They then execute Π on these inputs.

The output of Π is

$$t^m((x, x_2^*, \ldots, x_m^*), (y, y_2^*, \ldots, y_m^*)) = t(x, y) \bigwedge_{j=2}^{m} 1 = t(x, y).$$

The protocol uses at most k rounds and the same communication cost as Π. Thus, $D^k(t) \leq D^k(t^m)$.

Next, prove $D^k(t^m) \leq m \cdot D^k(t)$.

Let Π be a deterministic protocol computing t using at most k rounds and communication cost $D^k(t)$.

To compute t^m, Alice and Bob execute m independent copies of Π in parallel, one on each block (x_i, y_i).

In each round of the parallel protocol, the player who would send a message in a single execution of Π instead sends the bundled messages corresponding to what they would send in that round for each of the m independent executions (concatenated in a fixed order).

Since the blocks are disjoint and the executions independent, this correctly simulates all m copies. The number of rounds remains at most k, and the total number of bits communicated is at most m times the number of bits in a single execution of Π.

Thus, $D^k(t^m) \leq m \cdot D^k(t)$. □

F.19 Problem 19

F.19.1 Problem Statement

Theorem F.18 (Exercise 4.31)**.** *For any k, define the pointer jumping function PJ_k as follows: The input is a $2n$-vertex directed graph of out-degree 1. Bob holds the outgoing edges from vertices $1, \ldots, n$ of the graph, and Alice holds the outgoing edges from vertices $n+1, \ldots, 2n$ (that is $\Theta(n \log n)$ bits each). The value of PJ_k is the least significant bit of the k-th vertex reached by following the path starting at vertex 1. Use an argument from T_k to show that $R^k(PJ_k) = \Omega(n^{1/k}/\log^{k-1} n)$. In contrast, note that $D^{k,Bob}(PJ_k) \leq k \log n$.*

F.19.2 GPT

Proof. Let us first define the terms involved in the theorem. The pointer jumping function PJ_k is defined such that the input is a $2n$-vertex directed graph with out-degree 1. Bob holds the outgoing edges from vertices $1, \ldots, n$, and Alice holds the outgoing edges from vertices $n+1, \ldots, 2n$. The value of PJ_k is the least significant bit of the k-th vertex reached by following the path starting at vertex 1.

To prove the desired bound on the communication complexity, we will examine two quantities: $R^k(PJ_k)$ and $D^{k,\text{Bob}}(PJ_k)$.

1. Lower Bound on $R^k(PJ_k)$

First, we examine the randomized communication complexity $R^k(PJ_k)$, which refers to the minimum number of bits that need to be exchanged between Alice and Bob to compute the k-th vertex reached by following the path starting at vertex 1. The key observation is that the path in the graph is determined by the sequence of outgoing edges, and since the graph is a directed graph with out-degree 1, the sequence of vertices is uniquely determined by the outgoing edges held by Alice and Bob.

Each step in the path requires at least one bit of information from either Alice or Bob, and the sequence of paths taken is determined by the initial vertex and the edges that Alice and Bob hold. Since Alice and Bob are holding disjoint sets of edges, the communication complexity depends on how the graph structure influences the path taken.

From the perspective of the randomized communication complexity, it is known that the least significant bit of the k-th vertex reached requires communication about the path taken. This path involves k steps, and the number of bits required to determine the k-th vertex will grow with n, the number of vertices in the graph.

By applying the argument from T_k, it follows that the lower bound on the randomized communication complexity is given by:

$$R^k(PJ_k) = \Omega\left(\frac{n^{1/k}}{\log^{k-1} n}\right).$$

This bound arises because the number of distinct paths that need to be considered increases as the number of vertices grows, and the least significant bit of the k-th vertex requires communication that scales with the complexity of the graph.

2. Upper Bound on $D^{k,\text{Bob}}(PJ_k)$

Next, we examine the deterministic communication complexity $D^{k,\text{Bob}}(PJ_k)$, which represents the number of bits that Bob must send to Alice in a deterministic protocol to compute the k-th vertex reached by following the path starting at vertex 1. Since Bob knows the outgoing edges from vertices $1, \ldots, n$, and the path to the k-th vertex depends only on these edges, the communication complexity is directly related to the number of vertices that Bob must communicate in the protocol.

Since each bit of communication in the deterministic protocol corresponds to a distinct decision about the path, and there are k such decisions, the upper bound on the communication complexity is given by:

$$D^{k,\text{Bob}}(PJ_k) \leq k \log n.$$

This upper bound follows from the fact that each step in the path involves communication of $\log n$ bits, and there are at most k steps in the path.

Conclusion

We have shown that the randomized communication complexity satisfies:

$$R^k(PJ_k) = \Omega\left(\frac{n^{1/k}}{\log^{k-1} n}\right),$$

while the deterministic communication complexity satisfies:

$$D^{k,\text{Bob}}(PJ_k) \leq k \log n.$$

Thus, the theorem is proven. □

F.19.3 Claude

Proof. We prove the lower bound $R^k(PJ_k) = \Omega(n^{1/k}/\log^{k-1} n)$ by reduction from the k-round communication complexity of the tribes function T_k.

First, we recall the key properties of the tribes function T_k. The tribes function is defined on n bits organized into t tribes of size s each, where $n = ts$ and typically $s = \Theta(\log n)$ and $t = \Theta(n/\log n)$. The function outputs 1 if and only if there exists a tribe where all bits are 1. It is known that $R^k(T_k) = \Omega(n^{1/k}/\log^{k-1} n)$.

We now construct a reduction from T_k to PJ_k. Given an instance of T_k with n input bits distributed between Alice and Bob, we construct an instance of PJ_k as follows:

Consider a directed graph with $2n$ vertices. We organize the graph into k layers, where the path starting from vertex 1 will traverse through these layers. Each layer corresponds to one round of communication in the protocol for T_k.

The construction proceeds as follows: We embed the computation of T_k into the structure of the graph such that:

- The path from vertex 1 makes exactly k steps through the graph.
- At each step i (for $i = 1, \ldots, k$), the choice of which vertex to visit next depends on a portion of the input to T_k.
- The edges controlled by Alice (from vertices $n+1, \ldots, 2n$) encode Alice's bits in T_k.
- The edges controlled by Bob (from vertices $1, \ldots, n$) encode Bob's bits in T_k.
- After k steps, the final vertex reached has its least significant bit equal to the value of T_k on the given input.

More precisely, we partition the tribes into k groups. At round i of the path traversal, the next vertex visited is determined by examining approximately $n^{1/k}$ tribes. The structure ensures that:

If there exists a k-round protocol Π for PJ_k with communication cost c, then we can simulate a k-round protocol for T_k with the same communication cost c as follows:

Alice and Bob simulate the protocol Π on the constructed graph instance. Since the graph structure encodes the input to T_k, and since PJ_k outputs the correct value of T_k by construction, the simulation yields a valid protocol for T_k.

The key observation is that each message in the protocol for PJ_k corresponds to specifying which vertex is reached after some number of steps. Since the graph has $2n$ vertices, each message requires at most $\log(2n) = O(\log n)$ bits to specify a vertex. However, the protocol must distinguish between different paths through the graph, and the number of possible paths grows exponentially with the structure of the tribes.

By the lower bound for T_k, we have $R^k(T_k) = \Omega(n^{1/k}/\log^{k-1} n)$. Since we have constructed a reduction from T_k to PJ_k that preserves the number of rounds and does not increase the communication complexity by more than a constant factor, we conclude that:

$$R^k(PJ_k) = \Omega(n^{1/k}/\log^{k-1} n).$$

For the upper bound on deterministic communication with Bob speaking first, note that Bob can send the identity of the vertex reached after step 1 (using $\log n$ bits), then Alice sends the vertex after step 2 (using $\log n$ bits), and so on. After k messages of $\log n$ bits each, both parties know the final vertex and can compute its least significant bit. Thus $D^{k,\text{Bob}}(PJ_k) \leq k \log n$. □

F.19.4 Gemini

F.19.5 Grok

Proof. We first prove the upper bound $D^{k,\text{Bob}}(PJ_k) \leq k \log n$ (absorbing constants in the $\log n$ term, assuming suitable ceilings).

A deterministic protocol with Bob having the advantage is as follows: the players iteratively follow the path from the current vertex v, starting with $v = 1$.

The player owning the current v locally follows the path by applying their outgoing edges as long as the path remains in their domain or the k jumps are not completed.

When the path would enter the other player's domain, the current player sends the current v to the other player ($O(\log n)$ bits).

This requires at most k messages in the worst case (when the domain switches every jump), each of $O(\log n)$ bits, for total cost $O(k \log n)$.

The arrangement ensures Bob can handle consecutive jumps in his domain locally, bounding the cost when Bob has the advantage in starting and parity.

For the lower bound $R^k(PJ_k) = \Omega(n^{1/k}/\log^{k-1} n)$, we proceed by induction on k, adapting the argument from the lower bound for the function T_k (the analogous branching pointer jumping problem with explicit tree structure).

The base case and inductive step follow the standard recursive structure for round-communication tradeoffs in nested composition functions: a k-round protocol with small cost for size n would imply a $(k-1)$-round protocol with inappropriately small cost for a subsampled instance of size $\Theta(n/2^{O(c)})$, contradicting the induction hypothesis after balancing the branching factor $2^{O(c)} \approx n^{1/k}$ and accumulating the $\log n$ factors from encoding the "index" at each level.

The details follow the standard argument for such hierarchical lower bounds in communication complexity, yielding the stated bound. □

F.20 Problem 20

F.20.1 Problem Statement

Theorem F.19. *Show that* $R^k(\rho_{U_k}) \leq R^{k-1,Bob}(\rho_{U_k}) = O\left(\frac{n \log n}{k}\right)$.

F.20.2 GPT

Proof. We are tasked with proving that

$$R^k(\rho_{U_k}) \leq R^{k-1,\text{Bob}}(\rho_{U_k}) = O\left(\frac{n \log n}{k}\right).$$

To establish this result, we will proceed in several steps.

First, recall the definitions of $R^k(\rho_{U_k})$ and $R^{k-1,\text{Bob}}(\rho_{U_k})$. These represent certain complexity measures related to a process involving ρ_{U_k}, where k denotes the level of iteration and n the size of the system.

Step 1: Upper Bound on $R^k(\rho_{U_k})$

We begin by considering the recursive structure of $R^k(\rho_{U_k})$. Since $R^k(\rho_{U_k})$ involves a k-th level iteration, it is known from previous results that each iteration contributes a complexity term that is bounded by $O\left(\frac{n \log n}{k}\right)$ for sufficiently large k. This follows from the scaling behavior observed in similar recursive structures, where the complexity at level k is proportional to the total size of the system n, divided by the level of recursion k. Specifically,

$$R^k(\rho_{U_k}) = O\left(\frac{n \log n}{k}\right).$$

Thus, we have shown that

$$R^k(\rho_{U_k}) \leq O\left(\frac{n \log n}{k}\right).$$

Step 2: Relationship between $R^k(\rho_{U_k})$ and $R^{k-1,\text{Bob}}(\rho_{U_k})$

Next, we examine the relationship between $R^k(\rho_{U_k})$ and $R^{k-1,\text{Bob}}(\rho_{U_k})$. From the recursive structure, it is known that $R^{k-1,\text{Bob}}(\rho_{U_k})$ can be viewed as the complexity at level $k-1$, and thus by similar reasoning,

$$R^{k-1,\text{Bob}}(\rho_{U_k}) = O\left(\frac{n\log n}{k-1}\right).$$

Since $k \geq 2$, we have the inequality

$$O\left(\frac{n\log n}{k}\right) \leq O\left(\frac{n\log n}{k-1}\right),$$

which implies

$$R^k(\rho_{U_k}) \leq R^{k-1,\text{Bob}}(\rho_{U_k}).$$

Thus, we have shown the required inequality

$$R^k(\rho_{U_k}) \leq R^{k-1,\text{Bob}}(\rho_{U_k}).$$

Step 3: Conclusion

Combining the results from Step 1 and Step 2, we conclude that

$$R^k(\rho_{U_k}) \leq R^{k-1,\text{Bob}}(\rho_{U_k}) = O\left(\frac{n\log n}{k}\right).$$

This completes the proof. □

F.20.3 Claude

Proof. We prove the inequality $R^k(\rho_{U_k}) \leq R^{k-1,\text{Bob}}(\rho_{U_k}) = O\left(\frac{n\log n}{k}\right)$ in two parts.

Part 1: Showing $R^k(\rho_{U_k}) \leq R^{k-1,\textbf{Bob}}(\rho_{U_k})$.

By definition, $R^k(\rho_{U_k})$ denotes the k-round communication complexity of the protocol with state ρ_{U_k}, while $R^{k-1,\text{Bob}}(\rho_{U_k})$ denotes the $(k-1)$-round communication complexity where Bob sends the final message.

Any protocol achieving $R^{k-1,\text{Bob}}(\rho_{U_k})$ is also a valid k-round protocol, since a $(k-1)$-round protocol with Bob's final message can be viewed as a k-round protocol by potentially adding a null message from Alice. Therefore, the set of $(k-1)$-round protocols with Bob's final message contains all k-round protocols up to this equivalence, which implies

$$R^k(\rho_{U_k}) \leq R^{k-1,\text{Bob}}(\rho_{U_k}).$$

Part 2: Showing $R^{k-1,\textbf{Bob}}(\rho_{U_k}) = O\left(\frac{n\log n}{k}\right)$.

Consider the uniform distribution U_k over inputs of size n that require exactly k rounds of communication. For the state ρ_{U_k}, we analyze the communication required in a $(k-1)$-round protocol where Bob sends the final message.

By the information-theoretic argument, the total communication in an optimal protocol is bounded by the mutual information between the inputs and the transcript. For a k-round protocol on n-bit inputs with uniform distribution, we have

$$I(X;Y|T_{k-1}) \leq H(X|T_{k-1}),$$

where T_{k-1} represents the transcript after $k-1$ rounds.

By the round elimination lemma and standard information-theoretic bounds, after $k-1$ rounds of communication, the remaining uncertainty about the input is distributed among the k rounds. Using the pigeonhole principle, at least one round must communicate at most $\frac{1}{k}$ of the total information.

The total information to be communicated is $O(n)$ bits. By amortization over k rounds and accounting for the logarithmic overhead from encoding, we obtain

$$R^{k-1,\text{Bob}}(\rho_{U_k}) \leq \frac{n}{k} \cdot O(\log n) = O\left(\frac{n\log n}{k}\right).$$

The logarithmic factor arises from the need to encode messages efficiently and handle the protocol structure.

Combining both parts, we conclude that

$$R^k(\rho_{U_k}) \leq R^{k-1,\text{Bob}}(\rho_{U_k}) = O\left(\frac{n \log n}{k}\right).$$

□

F.20.4 Gemini

<think>**Defining the Proof Strategy**

I'm currently breaking down the LaTeX exercise statement. My focus is on clearly defining the role – a rigorous mathematical proof writer. The input, a specific LaTeX exercise, is under detailed analysis to grasp its core requirements. Initial steps involve identifying key variables and the overall objective.

Interpreting Mathematical Context

I've moved on to examining the mathematical statement's context. My focus is now the meaning of the variables and their relations. Decoding the notations is key, so I'm investigating the likely origin, possibly quantum information, and the implied task, potentially a "Hidden Matching" problem variant. Currently, I'm working to fully understand the notation's definition.

Refining the Approach

I'm now zeroing in on the specific Hidden Matching variant. Given the bound, I believe I can leverage known results and techniques. My focus is on adapting an existing proof structure to fit the exercise's precise constraints, especially the role and strict formatting. I'm exploring how to frame the solution within a "communication protocol" context.

Revisiting Proof Structure

I'm now back to the fundamental protocol construction, as the core problem is a standard communication complexity result, likely around Hidden Matching. Currently, I'm adapting known techniques to match the exercise, focusing on the appropriate protocol design and communication size bound. The focus is to precisely fulfill the strict LaTeX output constraints, including the proof environment and formatting.

Re-evaluating the Strategy

The core of the proof is the construction. Now, I'm focusing on adapting known techniques to match the exercise's constraints. I'm focusing on the communication protocol, aiming for the $O(\frac{n \log n}{k})$ bound. Currently, I'm verifying the role of the $\log n$ factor. The main challenge remains: precisely fulfilling the LaTeX output rules.

Formulating Protocol Steps

My strategy now involves constructing the $O(\frac{n \log n}{k})$ communication protocol. It seems I need Bob to start, and the core will be a communication-efficient Hidden Matching protocol. I'm focusing on the partitioning of indices, random sampling, and message sizes. Right now, I'm verifying the number of communication rounds and how the indices are going to be split.

Constructing the Communication Protocol

The next step is to detail the $O(\frac{n \log n}{k})$ communication protocol for the given setting. I'm currently constructing Bob's actions in round one. The strategy uses partitioning and targeted information exchange. The primary goal is to minimize the amount of data Bob must communicate, focusing on the Hidden Matching problem.

Protocol Decomposition

I'm now deep into designing the protocol, working to precisely understand how Alice and Bob will exchange information. My focus is on breaking down the problem into smaller communication steps. Currently, I'm examining alternative ways to structure Bob's initial message.

Revising Hidden Matching Protocol

I'm revisiting the core strategy for the communication protocol, focusing on the Hidden Matching problem and how to use the given $O(\frac{n \log n}{k})$ bound. Now, I'm thinking of how to partition the index space and then how to handle the communication of information between Alice and Bob. The goal is a concrete, implementable protocol.

Finalizing the Proof Structure

I'm now determining how to encode the indices efficiently. The goal is to minimize communication volume while adhering to the specified bound and LaTeX formatting. Right now, I'm working out the precise sampling strategy to minimize communication in each round, focusing on $O(\frac{n \log n}{k})$.

Reassessing the Bound

I'm now revisiting the $O(\frac{n \log n}{k})$ bound and the role of Bob starting. The strategy appears to involve Bob querying specific bits from Alice based on his matching. I'm focusing on how the partitioning of indices and probabilistic methods can achieve the desired communication cost, specifically the interplay between the indices and the number of rounds.

Finalizing Proof Structure

I'm now focused on the exact implementation of Bob's protocol steps and their LaTeX representation. My goal is to use partitioning to reduce communication volume, aiming for $O(\frac{n \log n}{k})$, while ensuring the output aligns with the format requirements. Right now, I'm working to describe how Bob, in the first round, can choose indices and request information from Alice.

Defining Protocol Parameters

I'm now focusing on parameterizing the communication protocol for the given bound. My goal is to clearly establish how Alice and Bob exchange information, including the number of rounds and data transferred. The $O(\frac{n \log n}{k})$ bound necessitates careful consideration of communication efficiency and the roles of both parties.

Reiterating Problem Analysis

I'm now consolidating the mathematical statement and my understanding of the relevant problem. I'm focusing on the nature of ρ_{U_k}, and the Hidden Matching context, and then mapping it to the output requirements. I'm verifying the potential protocol structure.

Reiterating the Core Task

I'm focused on grasping the specific instance of the Hidden Matching problem. My goal is to outline the protocol steps clearly, in order to prove the $O(\frac{n \log n}{k})$ bound. Currently, I am considering the roles of Alice and Bob and their interactions and messages, while trying to stay within the LaTeX code requirements.

Defining the Proof Steps

I'm now starting to outline the communication protocol in detail. My focus is to break it down into sequential steps, keeping Bob's actions in round one in mind. I'm carefully considering how to structure the messages and the data exchanged, aiming to fulfill the given bound.

Analyzing Protocol Structure

I'm now consolidating the protocol steps for the Hidden Matching variant and the $O(\frac{n \log n}{k})$ bound. Currently, my thought process is focusing on structuring the messages and verifying the overall communication cost. The core challenge is the exact parameterization of the protocol to meet all requirements.

Protocol Initialization

I'm now starting to draft the protocol, focusing on a communication-efficient approach. My aim is to define the roles of Alice and Bob and their interactions, and to detail the index partitioning and the size of the messages. I'm focusing on Bob's first message to Alice.

Constructing the Communication

I'm now detailing the steps, considering that Bob initiates the protocol. My focus is the role of partitioning and communication costs. I'm aiming for the target bound while accurately portraying the process within the LaTeX structure.

Reiterating the Core Steps

I'm focused on developing the protocol steps, assuming the standard Hidden Matching problem or a variant. My goal is to outline the interaction between Alice and Bob, ensuring the protocol achieves the desired bound. I am considering the index partitioning and message structure.

Defining Proof Structure

I'm now outlining a formal proof structure, emphasizing the "Bob-starts" aspect. My focus is on framing the problem, establishing the general inequality. Specifically, constructing a communication protocol to meet the required bound.

Outlining the Core Argument

I'm now consolidating the essential points for the Hidden Matching proof, aiming for conciseness. My focus is on breaking down the strategy into easily digestible, LaTeX-friendly steps. I am considering the key components for both the inequality and the $O(\frac{n \log n}{k})$ bound to produce an effective proof.

Formulating the Proof

I'm now consolidating the steps. My primary goal is to outline the protocol for the chosen problem, ensuring that it meets all of the performance criteria and formal LaTeX requirements. The construction of Bob's initial message is very important.

Protocol Drafting Begins

I'm now starting to craft the core of the proof, focusing on the Hidden Matching problem as a foundation. My goal is to rigorously build a communication protocol meeting the specified bound and formal LaTeX code requirements. I'm carefully constructing the protocol step-by-step.

Constructing the Core Proof

I'm now fully immersed in outlining the Hidden Matching communication protocol. My goal is precise and comprehensive LaTeX output. I'm focusing on ensuring Bob's initial moves, indexing, partitioning, and the critical role of randomization. This includes describing how the index space gets effectively handled.

Proof Implementation Underway

I'm now in the process of translating the protocol into precise LaTeX code. My focus is on ensuring the protocol adheres to the strict formatting requirements. Currently, I'm working to describe Bob's actions, including how he partitions the indices, selects the necessary data, and how the communication steps translate into concise mathematical expressions.

Codifying the Hidden Matching Proof

I'm now implementing the Hidden Matching protocol in LaTeX. My goal is to use partitioning and targeted information exchange to achieve the $O(\frac{n \log n}{k})$ bound. I'm carefully describing the communication steps in LaTeX, using mathematical notation and equations.

Formulating the Proof Structure

I'm now consolidating the various components of the proof into a cohesive structure, ready for LaTeX implementation. My goal is a concise and mathematically sound presentation of the communication protocol and related arguments, while keeping within the specified output restrictions. Specifically, I am refining the communication protocol for Bob, assuming a shared randomness/public coin scenario.

Proof Construction Begins

I'm now starting to craft the core of the proof, focusing on the Hidden Matching problem as a foundation. My goal is to rigorously build a communication protocol meeting the specified bound and formal LaTeX code requirements. I'm carefully constructing the protocol step-by-step.

Constructing Formal Arguments

I'm now fully immersed in developing the formal proof structure, which must be clearly defined and mathematically correct. My plan is to start with the trivial inequality and then outline a communication protocol. Right now, I'm verifying the basic principles.

</think>

Proof. We first establish the inequality $R^k(\rho_{U_k}) \leq R^{k-1,\text{Bob}}(\rho_{U_k})$. Let $\mathcal{P}$ be any protocol in the class defining $R^{k-1,\text{Bob}}(\rho_{U_k})$. By definition, in such a protocol, Bob initiates the communication, and the total number of messages exchanged is at most $k-1$. Consider the class of protocols defining $R^k(\rho_{U_k})$, where Alice initiates the communication and at most k messages are exchanged. Alice can simulate the protocol $\mathcal{P}$ by sending an empty initial message (or a dummy message of length 0) to Bob. Following this, Bob and Alice proceed exactly as in $\mathcal{P}$. The total number of messages in this simulated protocol is $1 + (k-1) = k$. Since the first message conveys no information, the communication cost is identical to that of $\mathcal{P}$. Thus, any protocol valid for $R^{k-1,\text{Bob}}(\rho_{U_k})$ is a valid protocol for $R^k(\rho_{U_k})$, implying

$$R^k(\rho_{U_k}) \leq R^{k-1,\text{Bob}}(\rho_{U_k}).$$

Next, we establish the upper bound $R^{k-1,\text{Bob}}(\rho_{U_k}) = O\left(\frac{n \log n}{k}\right)$. We consider the Hidden Matching problem instance typically associated with the state ρ_{U_k}. Let the input size be n. Alice holds a string

$x \in \{0,1\}^n$ and Bob holds a matching M on the indices $\{1,\dots,n\}$. Bob's goal is to output the parity $x_i \oplus x_j$ for some edge $(i,j) \in M$.

We construct a randomized classical protocol with one-way communication from Bob to Alice, followed by a response from Alice (which constitutes 2 messages, fitting within $k-1$ rounds for $k \geq 3$, or generally representing the dominant cost in the interaction).

Protocol Π:

1. Using public shared randomness, Alice and Bob agree on a family of random subsets of $\{1,\dots,n\}$.
2. Bob selects a subset $S \subset \{1,\dots,n\}$ of size $m = \lceil \frac{c \cdot n \log n}{k} \rceil$ for some constant c, such that with high probability, S contains both endpoints of at least one edge $(i,j) \in M$.
3. Bob sends the index (seed) of the chosen subset S to Alice. The cost of this message is $O(\log n)$.
4. Alice responds by sending the substring $x|_S$, consisting of the bits of x at the indices in S. The cost of this message is $m = O\left(\frac{n \log n}{k}\right)$.
5. Bob receives $x|_S$. Since S contains an edge $(i,j) \in M$, Bob extracts x_i and x_j and computes $x_i \oplus x_j$.

Analysis: The probability that a specific edge $(i,j) \in M$ has both endpoints in a random subset S of size m is approximately $(m/n)^2$. The expected number of edges captured by S is $|M| \cdot (m/n)^2 = \frac{n}{2} \cdot \left(\frac{cn \log n}{nk}\right)^2 \approx \frac{c^2 n (\log n)^2}{2k^2}$. By adjusting the constant c and the subset size slightly to account for the specific constraints of the Hidden Matching distribution ρ_{U_k} (where often k partitions are considered), we ensure the success probability is at least $2/3$. Specifically, for the standard tradeoff, partitioning the input into k blocks allows Bob to identify a relevant block with high probability. The communication cost is dominated by Alice's message:

$$\text{Cost}(\Pi) = O(\log n) + O\left(\frac{n \log n}{k}\right) = O\left(\frac{n \log n}{k}\right).$$

Since Bob speaks first, this protocol belongs to the class $R^{k-1,\text{Bob}}$ (specifically, it is a 2-round protocol, which satisfies the constraint for any $k \geq 3$; for general k, iterative block revelation yields the same asymptotic bound). Therefore,

$$R^{k-1,\text{Bob}}(\rho_{U_k}) = O\left(\frac{n \log n}{k}\right).$$

Combining this with the first inequality, the proof is complete. □

F.20.5 Grok

Proof. The inequality $R^k(\rho_{U_k}) \leq R^{k-1,\text{Bob}}(\rho_{U_k})$ is immediate from the definitions, as any protocol with at most k rounds can be transformed into a protocol with at most k rounds where Bob has the advantage (e.g., by having Alice send an empty message at the beginning if she would otherwise start, effectively making Bob the "advantaged" player without increasing the communication cost).

For the upper bound $R^{k-1,\text{Bob}}(\rho_{U_k}) = O\left(\frac{n \log n}{k}\right)$, we construct an explicit randomized protocol where Bob starts, using at most $k-1$ rounds, with total communication $O\left(\frac{n \log n}{k}\right)$.

The relation ρ_{U_k} is the carefully constructed "universal" relation (or the associated search problem) designed to exhibit a smooth round-communication tradeoff, based on a layered graph with exponentially growing layer sizes tuned to the parameter k.

The protocol exploits Bob's advantage as follows: the layers are assigned so that Bob owns the majority of the "heavy" (large) layers.

Bob locally computes the composed pointers for his large consecutive blocks of layers, which he can do without communication.

At each switch to an Alice-owned block, Bob sends the current position ID in the input to Alice's block, which has size $O(\frac{n}{k})$ due to the balanced grouping into effectively k super-blocks under the advantage.

The log of the position ID is $O(\log n)$, but since the effective number of switches is small under the advantage, or the positions sent are in small layers, the cost per message is small, but the grouping ensures the total information sent is $O(\frac{n \log n}{k})$.

Since there are at most $k-1$ rounds, the interaction allows resolving the small blocks interactively if needed, but the dominant cost is the composed descriptions for the balanced super-blocks, bounded by $O(\frac{n \log n}{k})$.

The randomness is used to randomize the assignment or the grouping to ensure the balancing with high probability, ensuring bounded error if necessary for the relation.

This yields the desired bound. □

F.21 Problem 21

F.21.1 Problem Statement

Theorem F.20 (Exercise 4.34)**.** *Prove that each of the functions* EQ*,* GT*,* IP*, and* DISJ *has an* $[\frac{n}{2}+1, \frac{n}{2}+1]$*-protocol. In contrast, prove that for some constant* c*, most functions* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *do not have an* $[n - c \log n, n - c \log n]$*-protocol.*

F.21.2 GPT

Proof. We will prove the two parts of the theorem: (1) that each of the functions EQ, GT, IP, and DISJ has an $[\frac{n}{2}+1, \frac{n}{2}+1]$-protocol, and (2) that for some constant c, most functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ do not have an $[n - c \log n, n - c \log n]$-protocol.

Part 1: Existence of an $[\frac{n}{2}+1, \frac{n}{2}+1]$-protocol for EQ, GT, IP, and DISJ

Let us first examine the function EQ. The function $\mathsf{EQ}(x, y)$ outputs 1 if and only if $x = y$. A straightforward protocol for EQ involves comparing corresponding bits of x and y using a pair of communicators. Each player sends one bit at a time, comparing the i-th bit of x with the i-th bit of y. Since a correct answer requires all bits of x and y to match, the protocol can be completed by communicating at most $\frac{n}{2}+1$ bits from each player, resulting in an $[\frac{n}{2}+1, \frac{n}{2}+1]$-protocol.

Next, consider the function $\mathsf{GT}(x, y)$, which outputs 1 if $x > y$ and 0 otherwise. A protocol for GT can similarly use a bit-by-bit comparison, with each player communicating the result of comparing the i-th bit of x with the i-th bit of y. Since the comparison is lexicographically ordered, once the first mismatch is found (indicating the result), the communication can terminate. The number of bits exchanged is still within $\frac{n}{2}+1$ for each player, thus satisfying the condition for an $[\frac{n}{2}+1, \frac{n}{2}+1]$-protocol.

For the function $\mathsf{IP}(x, y)$, which computes the inner product modulo 2, the players can exchange bits to calculate the sum of the product of corresponding bits modulo 2. This can be done with a protocol requiring at most $\frac{n}{2}+1$ bits from each player, following the same principles as the previous cases.

Lastly, for the function $\mathsf{DISJ}(x, y)$, which outputs 1 if there is at least one index i such that $x_i \neq y_i$, the players can compare the bits of x and y one by one and stop once they find a mismatch. This comparison can be performed with at most $\frac{n}{2}+1$ bits exchanged by each player.

Thus, all four functions EQ, GT, IP, and DISJ have an $[\frac{n}{2}+1, \frac{n}{2}+1]$-protocol.

Part 2: Non-existence of an $[n - c \log n, n - c \log n]$-protocol for most functions

Let us now consider the second part of the theorem. We wish to show that for some constant c, most functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ do not have an $[n - c \log n, n - c \log n]$-protocol.

To prove this, we consider the total number of possible functions from $\{0,1\}^n \times \{0,1\}^n$ to $\{0,1\}$, which is 2^{2^n}. A protocol for a function f with communication complexity $[n - c \log n, n - c \log n]$ would imply that the function is computable by a protocol that communicates at most $n - c \log n$ bits from each player.

We now argue that the number of functions that admit such a protocol is at most $2^{2(n - c \log n)}$. This is because for a protocol of length at most $n - c \log n$, there are at most $2^{n - c \log n}$ possible messages that can be communicated from each player. The total number of functions that can be computed with this amount of communication is therefore at most the number of possible message pairs, which is $2^{2(n - c \log n)}$.

However, the total number of functions from $\{0,1\}^n \times \{0,1\}^n$ to $\{0,1\}$ is 2^{2^n}. As n grows large, the ratio of the number of functions that can be computed by a protocol with communication complexity $[n - c\log n, n - c\log n]$ to the total number of functions approaches zero. Therefore, most functions do not have an $[n - c\log n, n - c\log n]$-protocol.

This completes the proof. □

F.21.3 Claude

Proof. We prove both parts of the theorem.

Part 1: Upper bounds for specific functions.

We first establish that each of the functions EQ, GT, IP, and DISJ has an $[\frac{n}{2}+1, \frac{n}{2}+1]$-protocol.

Recall that an $[s,t]$-protocol is a communication protocol where Alice sends a message of at most s bits and Bob sends a message of at most t bits.

For each function, we exhibit such a protocol:

Function EQ: Alice has input $x \in \{0,1\}^n$ and Bob has input $y \in \{0,1\}^n$. The function $\mathsf{EQ}(x,y) = 1$ if and only if $x = y$.

Protocol: Alice sends the first $\frac{n}{2}+1$ bits of x to Bob. Bob compares these bits with the corresponding bits of y. If they differ, Bob outputs 0. Otherwise, Bob sends the last $\frac{n}{2}$ bits of y to Alice (note that $\frac{n}{2} \leq \frac{n}{2}+1$ for $n \geq 0$). Alice compares these with her last $\frac{n}{2}$ bits and outputs 1 if they match, 0 otherwise. This is an $[\frac{n}{2}+1, \frac{n}{2}+1]$-protocol.

Function GT: Interpreting x and y as binary numbers, $\mathsf{GT}(x,y) = 1$ if and only if $x > y$.

Protocol: Alice sends the first $\frac{n}{2}+1$ bits of x to Bob. Bob compares these with his first $\frac{n}{2}+1$ bits. If $x_{1..\frac{n}{2}+1} > y_{1..\frac{n}{2}+1}$, Bob outputs 1. If $x_{1..\frac{n}{2}+1} < y_{1..\frac{n}{2}+1}$, Bob outputs 0. If they are equal, Bob sends his last $\frac{n}{2}$ bits to Alice, who determines whether $x > y$ by comparing the suffixes. This is an $[\frac{n}{2}+1, \frac{n}{2}+1]$-protocol.

Function IP: The inner product function $\mathsf{IP}(x,y) = \sum_{i=1}^{n} x_i y_i \pmod 2$.

Protocol: Alice computes $a = \sum_{i=1}^{\frac{n}{2}+1} x_i y_i \pmod 2$ by... wait, Alice does not know y_i. Instead, Alice sends $x_1, \ldots, x_{\frac{n}{2}+1}$ to Bob. Bob computes $a = \sum_{i=1}^{\frac{n}{2}+1} x_i y_i \pmod 2$. Bob sends $y_{\frac{n}{2}+2}, \ldots, y_n$ to Alice (which is $\frac{n}{2}$ bits when n is even, at most $\frac{n}{2}+1$ bits). Alice computes $b = \sum_{i=\frac{n}{2}+2}^{n} x_i y_i \pmod 2$ and outputs $a \oplus b$. This is an $[\frac{n}{2}+1, \frac{n}{2}+1]$-protocol.

Function DISJ: $\mathsf{DISJ}(x,y) = 1$ if and only if x and y are disjoint, i.e., for all i, $x_i \cdot y_i = 0$.

Protocol: Alice sends the first $\frac{n}{2}+1$ bits of x to Bob. Bob checks if $x_i \cdot y_i = 1$ for any $i \leq \frac{n}{2}+1$. If yes, Bob outputs 0. Otherwise, Bob sends the last $\frac{n}{2}$ bits of y to Alice. Alice checks if $x_i \cdot y_i = 1$ for any $i > \frac{n}{2}+1$ and outputs 0 if yes, 1 otherwise. This is an $[\frac{n}{2}+1, \frac{n}{2}+1]$-protocol.

Part 2: Lower bound for most functions.

We now prove that for some constant $c > 0$, most functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ do not have an $[n - c\log n, n - c\log n]$-protocol.

The total number of functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is $2^{2^{2n}}$.

For an $[s,t]$-protocol, Alice can send at most 2^s distinct messages and Bob can send at most 2^t distinct messages. The protocol partitions the input space into at most $2^s \cdot 2^t$ rectangles (each determined by Alice's message and Bob's message). Since the function must be constant on each rectangle, the protocol can compute at most $2^{2^s \cdot 2^t}$ distinct functions.

For $s = t = n - c\log n$, the number of computable functions is at most:

$$2^{2^{2(n-c\log n)}} = 2^{2^{2n-2c\log n}} = 2^{2^{2n}/n^{2c}} = 2^{2^{2n}/2^{2c\log n}} = 2^{2^{2n}/2^{\log n^{2c}}}$$

For $c = 1$, we have $2^{2^{2n}/n^2}$, which is much smaller than $2^{2^{2n}}$ for large n. More precisely, the fraction of functions with such a protocol is at most $2^{2^{2n}/n^2 - 2^{2n}} = 2^{-2^{2n}(1-1/n^2)} \to 0$ as $n \to \infty$.

Therefore, for sufficiently large n and appropriate constant c, most functions do not have an $[n - c\log n, n - c\log n]$-protocol. □

F.21.4 Gemini

Proof. We interpret an $[a, b]$-protocol as a deterministic interactive protocol where Alice sends at most a bits and Bob sends at most b bits in total throughout the interaction.

Part 1: Protocols for specific functions. We construct $[n/2 + 1, n/2 + 1]$-protocols for EQ, GT, IP, and DISJ on n-bit inputs x, y. We assume n is even (if n is odd, the bounds hold with ceilings). Let $x = (x_1, x_2)$ and $y = (y_1, y_2)$ where $x_1, y_1 \in \{0,1\}^{n/2}$ and $x_2, y_2 \in \{0,1\}^{n/2}$.

Protocol for EQ: 1. Alice sends x_1 to Bob. (Cost: $n/2$) 2. Bob sends y_2 to Alice. (Cost: $n/2$) 3. Alice computes $v_A = 1$ if $x_2 = y_2$, else 0. Alice sends v_A to Bob. (Cost: 1) 4. Bob computes $v_B = 1$ if $x_1 = y_1$, else 0. Bob sends v_B to Alice. (Cost: 1) 5. Both parties output $v_A \wedge v_B$. Total Alice: $n/2 + 1$. Total Bob: $n/2 + 1$. Validity: $\mathsf{EQ}(x, y) = (x_1 = y_1) \wedge (x_2 = y_2)$.

Protocol for IP: Recall $\mathsf{IP}(x, y) = (\sum x_i y_i) \pmod 2 = \mathsf{IP}(x_1, y_1) \oplus \mathsf{IP}(x_2, y_2)$. 1. Alice sends x_1. 2. Bob sends y_2. 3. Alice sends $v_A = \mathsf{IP}(x_2, y_2)$. 4. Bob sends $v_B = \mathsf{IP}(x_1, y_1)$. 5. Output $v_A \oplus v_B$. Total costs satisfy the bound.

Protocol for DISJ: $\mathsf{DISJ}(x, y) = 1 \iff \forall i, x_i y_i = 0$. This is equivalent to $\mathsf{DISJ}(x_1, y_1) \wedge \mathsf{DISJ}(x_2, y_2)$. The protocol is identical to EQ, replacing local checks with disjointness checks and the final operation with AND.

Protocol for GT: $\mathsf{GT}(x, y) = 1$ if $x > y$. This condition is $(x_1 > y_1) \vee (x_1 = y_1 \wedge x_2 > y_2)$. 1. Alice sends x_1. 2. Bob sends y_2. 3. Alice computes $v_A = 1$ if $x_2 > y_2$, else 0. She sends v_A. 4. Bob computes the result: If $x_1 > y_1$, output 1. If $x_1 < y_1$, output 0. If $x_1 = y_1$, output v_A. Bob sends the result bit to Alice. Total Alice: $n/2 + 1$. Total Bob: $n/2 + 1$.

Part 2: Lower bound for most functions. Let $K = n - c \log n$ for a sufficiently large constant c. Consider the set of all boolean functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. The total number of such functions is $2^{2^{2n}}$. We bound the number of functions computable by any $[K, K]$-protocol.

Fix a protocol Π where Alice sends $\leq K$ bits and Bob sends $\leq K$ bits. Let T denote the transcript of the communication. The total length of T is at most $2K$. The protocol partitions the input space into combinatorial rectangles $\{R_t\}_t$ corresponding to each transcript t. Since the protocol is deterministic, the output on R_t is a constant $v_t \in \{0,1\}$. However, for the protocol to define a function, the parties must be able to determine the output. Consider the final state. Since Alice sends at most K bits in total, for any fixed input y of Bob, the number of possible sequences of messages Bob receives from Alice is at most 2^K. Let S_y be the set of x's consistent with a transcript. If Bob outputs the value, he must distinguish the row behaviors restricted to S_y. Specifically, consider the constraint that Alice sends $\leq K$ bits. This implies that for any fixed y, the rows of the communication matrix M_f are partitioned into at most 2^K sets based on Alice's messages (which may depend on y). For a random function, we argue that no such protocol exists with high probability. Consider a specific input (x, y). Bob must determine $f(x, y)$. Bob knows y and the messages from Alice, which amount to K bits. Let M_A be Alice's message sequence. M_A is a function of x (and Bob's messages, which depend on y). Given y, M_A can take at most 2^K values as x varies. Thus, for a fixed y, the domain $\{0,1\}^n$ of x is partitioned into 2^K sets $X_{1,y}, \ldots, X_{L,y}$. Bob outputs a constant value for each set. This implies that for every y, the function $f(\cdot, y)$ must be constant on each $X_{i,y}$. Since $K = n - c \log n$, the average size of such a set is $2^{n-K} = 2^{c \log n} = n^c$. For a random function, the probability that $f(\cdot, y)$ is constant on a specific set of size S is $2 \cdot 2^{-S}$. Let $S = n^c$. The probability that a random function is consistent with a *fixed* partition for a fixed y is bounded. However, the partitions depend on the protocol. The number of protocols is roughly $2^{2^{2K}}$? No, much smaller. Alternative counting: Alice sends $\leq K$ bits. Bob sends $\leq K$ bits. Suppose Bob computes the result. For Bob to compute $f(x, y)$ correctly, he must know x "sufficiently". Information theoretically, Alice sends K bits about x. Bob has $H(x|\text{view}) \geq n - K = c \log n$. This suggests there is a set of $2^{c \log n}$ candidates for x indistinguishable to Bob. For a random function, the values $f(x, y)$ on this set are uniformly random. The probability that they are all equal (allowing Bob to output correctly) is $2 \cdot 2^{-2^{c \log n}} = 2^{-n^c + 1}$. This error probability is for a fixed y and fixed transcript view. Summing over all y and all protocols, for large c, the fraction of functions admitting such a protocol is:

$$\frac{\text{Functions with protocol}}{\text{Total functions}} \leq \text{Number of Protocols} \times 2^{2n} \times 2^{-n^c}.$$

The number of protocols is roughly bounded by $2^{\text{poly}(n)}$. The term 2^{-n^c} dominates (doubly exponential decay vs exponential). Thus, as $n \to \infty$, the fraction of functions having an $[n - c\log n, n - c\log n]$-protocol tends to 0. □

F.21.5 Grok

Proof. An $[a, b]$-protocol for a function $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is a nondeterministic simultaneous-message protocol with Alice sending an a-bit message and Bob sending a b-bit message such that there exists a set of accepting message pairs with the corresponding rectangles being 1-monochromatic and their union covering exactly $f^{-1}(1)$. The existence of such a protocol is equivalent to $C^1(f) \le 2^{a+b}$.

We first show that EQ, GT, IP, and DISJ each admit an $\left[\frac{n}{2}+1, \frac{n}{2}+1\right]$-protocol by constructing covers with at most 2^n 1-monochromatic rectangles (noting that $2^n \le 2^{n+2}/4 \le 2^{n+2}$ and $\left(\frac{n}{2}+1\right)+\left(\frac{n}{2}+1\right) = n+2$).

For $\mathsf{EQ}(x,y) = 1$ if and only if $x = y$, consider the 2^n rectangles $R_z = \{z\} \times \{z\}$ for each $z \in \{0,1\}^n$. Each R_z is 1-monochromatic, and their union is exactly $\mathsf{EQ}^{-1}(1)$.

For $\mathsf{GT}(x,y) = 1$ if and only if $x > y$ (interpreting x, y as integers), consider the 2^n rectangles $R_y = \{x \mid x > y\} \times \{y\}$ for each $y \in \{0,1\}^n$. Each R_y is 1-monochromatic, and their union is exactly $\mathsf{GT}^{-1}(1)$.

For $\mathsf{IP}(x,y) = \langle x, y\rangle \pmod 2 = 1$, consider the 2^n rectangles $R_y = \{x \mid \langle x, y\rangle = 1\} \times \{y\}$ for each $y \in \{0,1\}^n$. Each R_y is 1-monochromatic (of size 2^{n-1}), and their union is exactly $\mathsf{IP}^{-1}(1)$.

For $\mathsf{DISJ}(x,y) = 1$ if and only if $x \wedge y = 0$ (no common 1), consider the 2^n rectangles $R_y = \{x \mid x \wedge y = 0\} \times \{y\}$ for each $y \in \{0,1\}^n$. Each R_y is 1-monochromatic, and their union is exactly $\mathsf{DISJ}^{-1}(1)$.

In each case, $C^1(f) \le 2^n \le 2^{n+2}$, so an $\left[\frac{n}{2}+1, \frac{n}{2}+1\right]$-protocol exists (by assigning distinct $(n+2)$-bit indices to the rectangles and splitting each index into two parts of length $\frac{n}{2}+1$).

For the contrast, fix a constant $c > 2$ (to be specified). Consider $[n - c\log_2 n, n - c\log_2 n]$-protocols (assuming n large so the lengths are positive). Let $r = s = n - c\log_2 n$, so $r + s = 2n - 2c\log_2 n$.

A protocol is specified by Alice's nondeterministic message choices (for each of 2^n inputs x, a subset of the 2^r messages, giving at most $(2^{2^r})^{2^n} = 2^{2^n \cdot 2^r}$ possibilities), similarly for Bob ($2^{2^n \cdot 2^s}$), and the accepting pairs (at most $2^{2^{r+s}}$).

The total number of protocols is at most $2^{2^n(2^r+2^s)+2^{r+s}} = 2^{O(2^{2n-c\log_2 n})}$.

Each protocol correctly computes at most one function (the indicator function of the union of its accepting rectangles).

Thus, the number of functions admitting such a protocol is at most $2^{O(2^{2n-c\log_2 n})}$.

The total number of functions is $2^{2^{2n}}$.

Since $2^{2n-c\log_2 n} = 2^{2n}/n^c$ and $c > 2$ suffices to make $O(2^{2n-c\log_2 n}) < 2^{2n} - 1$ for large n, we have $2^{O(2^{2n-c\log_2 n})} \ll 2^{2^{2n}}$.

Hence, for sufficiently large c, most functions do not admit an $[n - c\log_2 n, n - c\log_2 n]$-protocol. □

F.22 Problem 22

F.22.1 Problem Statement

Theorem F.21 (Exercise 4.37)**.** *Let f be a (μ, v)-rich function. If f has an $[a, b]$-randomized protocol with one-sided error ε, then f contains a 1-monochromatic rectangle of dimensions at least $\frac{1-\varepsilon}{2^a} \cdot \frac{\mu}{2^b} \times \frac{1-\varepsilon}{2^b} \cdot \frac{v}{2^a}$.*

F.22.2 GPT

Proof. Let f be a (μ, v)-rich function and assume that f has an $[a, b]$-randomized protocol with one-sided error ε. Our goal is to show that f contains a 1-monochromatic rectangle with dimensions at least

$$\frac{1-\varepsilon}{2^a} \cdot \frac{\mu}{2^b} \times \frac{1-\varepsilon}{2^b} \cdot \frac{v}{2^a}.$$

We begin by noting that a (μ, v)-rich function f satisfies the property that for any large enough region in its domain, the proportion of inputs that map to a particular output is at least μ in one direction and v in the other direction.

Let the domain of f be $[n] \times [m]$ for some n and m, and let us consider the $[a, b]$-randomized protocol with one-sided error ε. By definition, the protocol gives a distribution over the inputs such that the probability of producing an incorrect output is at most ε, and the probability of producing a correct output is at least $1 - \varepsilon$. This protocol partitions the input space into regions corresponding to the possible outputs.

Now, consider the set of all rectangles in the domain of f. Each rectangle corresponds to a set of inputs that are mapped by the protocol to a particular output. Since f has one-sided error, we know that the portion of the domain where the protocol outputs the correct value is at least $1 - \varepsilon$. In particular, we can select a 1-monochromatic rectangle that contains a sufficiently large proportion of the inputs.

By the (μ, v)-richness of f, this rectangle must have dimensions that reflect the minimum density of correct outputs in both directions. Specifically, since the function is (μ, v)-rich, there must be a region in the domain where the proportion of correct outputs is at least $\frac{\mu}{2^b}$ in one direction and at least $\frac{v}{2^a}$ in the other direction. The size of this region can be bounded as follows.

Let R be the 1-monochromatic rectangle we are considering. The number of rows in R is at least $\frac{1-\varepsilon}{2^a} \cdot \mu$ and the number of columns is at least $\frac{1-\varepsilon}{2^b} \cdot v$, since the protocol guarantees a correct output for at least this fraction of inputs in each direction. Thus, the dimensions of the rectangle are at least

$$\frac{1-\varepsilon}{2^a} \cdot \frac{\mu}{2^b} \times \frac{1-\varepsilon}{2^b} \cdot \frac{v}{2^a}.$$

Therefore, the function f contains a 1-monochromatic rectangle of the required dimensions, completing the proof. □

F.22.3 Claude

Proof. Let $f : X \times Y \to \{0, 1\}$ be a (μ, v)-rich function, meaning there exists a 1-monochromatic rectangle $R_0 \subseteq X \times Y$ with $|R_0| \geq \mu \cdot v$. Suppose f has an $[a, b]$-randomized protocol $\mathcal{P}$ with one-sided error ε on inputs where $f(x, y) = 1$.

By definition of an $[a, b]$-randomized protocol, Alice sends at most a bits and Bob sends at most b bits. The protocol uses shared randomness $r \in \mathcal{R}$ where $|\mathcal{R}| \leq 2^a \cdot 2^b$.

For each random string $r \in \mathcal{R}$, the deterministic protocol $\mathcal{P}_r$ partitions $X \times Y$ into at most $2^a \cdot 2^b$ rectangles (corresponding to all possible transcripts).

Since the protocol has one-sided error ε, for any $(x, y) \in R_0$ (where $f(x, y) = 1$), we have:

$$\Pr_r[\mathcal{P}_r(x, y) = 1] \geq 1 - \varepsilon$$

For each $r \in \mathcal{R}$, let $R_0^r = \{(x, y) \in R_0 : \mathcal{P}_r(x, y) = 1\}$. Then:

$$\sum_{r \in \mathcal{R}} |R_0^r| = \sum_{(x,y) \in R_0} |\{r : \mathcal{P}_r(x, y) = 1\}| \geq (1 - \varepsilon) \cdot |\mathcal{R}| \cdot |R_0| \geq (1 - \varepsilon) \cdot |\mathcal{R}| \cdot \mu v$$

By the pigeonhole principle, there exists some $r^* \in \mathcal{R}$ such that:

$$|R_0^{r^*}| \geq \frac{(1 - \varepsilon) \cdot |\mathcal{R}| \cdot \mu v}{|\mathcal{R}|} = (1 - \varepsilon)\mu v$$

The set $R_0^{r^*}$ is contained in some 1-monochromatic rectangle R^* determined by the protocol $\mathcal{P}_{r^*}$ (the rectangle corresponding to a transcript that outputs 1).

Since $\mathcal{P}_{r^*}$ uses at most a bits from Alice and b bits from Bob, the rectangle R^* is a product of at most 2^a possible sets in X (determined by Alice's message) and at most 2^b possible sets in Y (determined by Bob's message).

Let $R^* = A^* \times B^*$ where $A^* \subseteq X$ and $B^* \subseteq Y$. Since $R_0 = A_0 \times B_0$ with $|A_0| \geq \mu$ and $|B_0| \geq v$, and $R_0^{r^*} \subseteq R_0 \cap R^*$, we have:

$$|R_0^{r^*}| = |A_0 \cap A^*| \cdot |B_0 \cap B^*|$$

The protocol partitions A_0 into at most 2^b parts (based on Bob's responses) and B_0 into at most 2^a parts (based on Alice's messages). By averaging, there exist sets $A' \subseteq A_0$ and $B' \subseteq B_0$ such that:

$$|A'| \geq \frac{|A_0|}{2^b} \geq \frac{\mu}{2^b}, \quad |B'| \geq \frac{|B_0|}{2^a} \geq \frac{v}{2^a}$$

and $A' \times B' \subseteq R^*$ is 1-monochromatic. Moreover, from the error bound:

$$|A'| \geq \frac{1-\varepsilon}{2^b} \cdot \mu, \quad |B'| \geq \frac{1-\varepsilon}{2^a} \cdot v$$

Therefore, f contains a 1-monochromatic rectangle of dimensions at least $\frac{1-\varepsilon}{2^a} \cdot \frac{\mu}{2^b} \times \frac{1-\varepsilon}{2^b} \cdot \frac{v}{2^a}$. □

F.22.4 Gemini

Proof. Let $f : X \times Y \to \{0,1\}$ be a (μ, v)-rich function. By definition, there exists a subset of rows $U \subseteq X$ with $|U| \geq \mu$ such that for every $x \in U$, the number of 1-entries is at least v. Let $S = \{(x,y) \in U \times Y \mid f(x,y) = 1\}$. The size of S satisfies $|S| \geq \mu v$.

Let Π be a randomized $[a,b]$-protocol for f with one-sided error ε. This protocol can be viewed as a probability distribution $\mathcal{D}$ over deterministic protocols $\{P_r\}$. Since the error is one-sided, for any (x,y) where $f(x,y) = 0$, $\Pr[\Pi(x,y) = 1] = 0$. For (x,y) where $f(x,y) = 1$, $\Pr[\Pi(x,y) = 1] \geq 1 - \varepsilon$. Considering the set S of 1-inputs, the expected number of inputs in S accepted by the protocol is:

$$\mathbb{E}_{P \sim \mathcal{D}}[|P^{-1}(1) \cap S|] = \sum_{(x,y) \in S} \Pr[\Pi(x,y) = 1] \geq \sum_{(x,y) \in S} (1-\varepsilon) = (1-\varepsilon)|S| \geq (1-\varepsilon)\mu v.$$

By the probabilistic method, there exists a deterministic protocol P in the support of $\mathcal{D}$ such that:

$$|P^{-1}(1) \cap S| \geq (1-\varepsilon)\mu v.$$

Let $S_P = P^{-1}(1) \cap S$. Since the error is one-sided, any input accepted by P must be a 1-input for f. Thus, $S_P \subseteq f^{-1}(1)$.

The deterministic protocol P uses at most a bits of communication from Alice and b bits from Bob. Therefore, P partitions the input space $X \times Y$ into at most 2^{a+b} disjoint combinatorial rectangles $\{R_i\}_{i \in I}$. Since S_P is the set of accepted inputs in S, and P is constant on each rectangle, S_P is the union of the intersections of these accepting rectangles with S:

$$S_P = \bigcup_{i \in I_{acc}} (R_i \cap S),$$

where I_{acc} is the set of indices of 1-monochromatic rectangles in P. The sum of the sizes is:

$$\sum_{i \in I_{acc}} |R_i \cap S| = |S_P| \geq (1-\varepsilon)\mu v.$$

Since the number of rectangles is at most 2^{a+b}, by the averaging principle, there exists a rectangle $R = A \times B$ such that:

$$|R \cap S| \geq \frac{(1-\varepsilon)\mu v}{2^{a+b}}.$$

This rectangle R is 1-monochromatic for P, and since P has one-sided error, R is 1-monochromatic for f. We now bound the dimensions of $R = A \times B$. The set $R \cap S$ consists of 1-entries located in R. 1. Bounding $|A|$: For any row $x \in A$, the number of 1-entries in the row $(x,y) \in R \cap S$ is at most v, because S contains only entries from rows with total Hamming weight at least v (and clearly the intersection with B cannot exceed the row weight). More precisely, the number of elements in $R \cap S$ cannot exceed $|A|$ times the maximum number of 1s in a row of S (which is v, or even $|Y|$, but bounded by v in the construction of rich sets implicitly or simply via $|R \cap S| \leq |A| \cdot v$ if we consider the density). Strictly, S is defined as the set of *all* 1s in the rows U. Thus for $x \in U$, the number of 1s is $\geq v$. However, the mass $|R \cap S|$ is contained in $A \times B$. We use the bounds:

$$|R \cap S| \leq |A \cap U| \cdot v \leq |A| \cdot v.$$

This implies:

$$|A| \geq \frac{|R \cap S|}{v} \geq \frac{(1-\varepsilon)\mu v}{v 2^{a+b}} = \frac{(1-\varepsilon)\mu}{2^{a+b}} = \frac{1-\varepsilon}{2^a} \cdot \frac{\mu}{2^b}.$$

2. Bounding $|B|$: The set $R \cap S$ is contained in $(A \cap U) \times B$. The number of rows contributing to the intersection is at most $|U| = \mu$. Thus:

$$|R \cap S| \leq |U| \cdot |B| = \mu \cdot |B|.$$

This implies:

$$|B| \geq \frac{|R \cap S|}{\mu} \geq \frac{(1-\varepsilon)\mu v}{\mu 2^{a+b}} = \frac{(1-\varepsilon)v}{2^{a+b}} = \frac{1-\varepsilon}{2^b} \cdot \frac{v}{2^a}.$$

Thus, the rectangle R is 1-monochromatic and has dimensions at least $\frac{1-\varepsilon}{2^a} \cdot \frac{\mu}{2^b} \times \frac{1-\varepsilon}{2^b} \cdot \frac{v}{2^a}$. □

F.22.5 Grok

Proof. Assume the domain is $X \times Y$, with $|X| = m$, $|Y| = l$. Without loss of generality, assume μ, ν are fractions, and the bounds are in terms of these fractions times the domain sizes.

The protocol is a private-coin randomized simultaneous protocol with one-sided error ε. Alice's messages are from $[2^a]$, Bob's from $[2^b]$, with fixed accepting set $S \subseteq [2^a] \times [2^b]$.

For each $x \in X$, let $p_{x,m_a} = \Pr[m_a \text{ sent on } x]$. Similarly, $q_{y,m_b} = \Pr[m_b \text{ sent on } y]$.

For $f(x,y) = 0$, $\sum_{(m_a,m_b)\in S} p_{x,m_a} q_{y,m_b} = 0$.

For $f(x,y) = 1$, the sum $\geq 1 - \varepsilon$.

Let $A \subseteq X$ with $|A| \geq \mu m$, and for each $x \in A$, $|Y_x| \geq \nu l$, where $Y_x = \{y \mid f(x,y) = 1\}$.

The sum over m_a of $\sum_{x\in A} p_{x,m_a} = |A| \geq \mu m$.

Thus, there exists m_a with $\sum_{x\in A} p_{x,m_a} \geq \mu m / 2^a$.

Let $S = \{x \in A \mid p_{x,m_a} > 0\}$. Since $p_{x,m_a} \leq 1$, $|S| \geq \sum_{x\in A} p_{x,m_a} \geq \mu m / 2^a$.

Let $Q = \{m_b \mid (m_a, m_b) \in S\}$.

Let $r_y = \sum_{m_b \in Q} q_{y,m_b}$.

For each $x \in S$, the average over $y \in Y_x$ of $p_{x,m_a} r_y + (1 - p_{x,m_a}) s_y \geq 1 - \varepsilon$, where $s_y \leq 1$ is the conditional accept probability given not m_a.

Thus, $p_{x,m_a} \cdot (\text{avg } r_y) + (1 - p_{x,m_a}) \geq 1 - \varepsilon$, so avg $r_y \geq 1 - \varepsilon / p_{x,m_a}$.

The average over $x \in S$ of avg $r_y \geq 1 - (\varepsilon/|S|) \sum_{x\in S} 1/p_{x,m_a}$.

By Cauchy-Schwarz, $\sum_{x\in S} 1/p_{x,m_a} \geq |S|^2 / \sum_{x\in S} p_{x,m_a}$.

Thus, average $\geq 1 - \varepsilon |S| / \sum_{x\in S} p_{x,m_a} \geq 1 - \varepsilon$, since $|S| / \sum p \leq 1$.

The average over $x \in S$ of avg $r_y = (1/|S|) \sum_y r_y \cdot (1/|S|) \sum_{x\in S} f(x,y)/|Y_x| \leq (1/|S|) \sum_y r_y \cdot (1/\nu l) \sum_{x\in S} f(x,y) \leq (1/|S|) \sum_y r_y \cdot (1/\nu l)|S| = \sum_y r_y/(\nu l)$.

Thus, $\sum_y r_y/(\nu l) \geq 1 - \varepsilon$, so $\sum_y r_y \geq (1-\varepsilon)\nu l$.

Then, average over $m_b \in Q$ of $\sum_y q_{y,m_b} \geq \sum_y r_y/|Q| \geq (1-\varepsilon)\nu l / 2^b$.

Thus, there exists $m_b \in Q$ with $\sum_y q_{y,m_b} \geq (1-\varepsilon)\nu l / 2^b$.

Let $T = \{y \mid q_{y,m_b} > 0\}$. Then $|T| \geq \sum_y q_{y,m_b} \geq (1-\varepsilon)\nu l / 2^b$.

The rectangle $S \times T$ is 1-monochromatic, since $(m_a, m_b) \in S$, and the supports are 1-mono for accepting pairs.

By symmetry, the bound holds in the other direction, and the theorem states a bound with $(1 - \varepsilon)$ in both dimensions.

□

F.23 Problem 23

F.23.1 Problem Statement

Theorem F.22 (Exercise 4.39)**.** *Let $DISJ_{k,\ell}$ be the disjointness function where Alice's set is of size k and Bob's set is of size ℓ (both sets are subsets of $\{1, \ldots, n\}$). Assume $k < \ell < n$. Then:*

(i) There exists a $[O(k \log \ell), O(k \log n)]$ protocol for $DISJ_{k,\ell}$.

(ii) If there exists an $[a, b]$-protocol for $DISJ_{k,\ell}$ where $a = o(k \log \ell)$, then $b = \ell^{1-o(1)}$.

F.23.2 GPT

Proof. We are tasked with proving two parts of the theorem related to the disjointness function $\text{DISJ}_{k,\ell}$, where Alice has a set of size k and Bob has a set of size ℓ.

Part (i): We begin by proving the existence of a protocol for $\text{DISJ}_{k,\ell}$ with communication complexity in the range $[O(k \log \ell), O(k \log n)]$. The disjointness function is defined as follows: Alice holds a set $A \subseteq \{1, \ldots, n\}$ with $|A| = k$, and Bob holds a set $B \subseteq \{1, \ldots, n\}$ with $|B| = \ell$. The goal of the protocol is to determine whether $A \cap B = \emptyset$.

First, we consider the following approach for designing the communication protocol. Alice will send a message to Bob that contains information about her set A. The most efficient way to represent her set is by sending a succinct representation, such as a binary string of length n, where the i-th bit is 1 if $i \in A$ and 0 otherwise. This message has length n, which is clearly too large for a protocol of small communication complexity. Therefore, we focus on a more efficient representation.

Using a **hashing technique**, Alice can hash the elements of her set A into a smaller set of indices. This approach allows Alice to send a small, compressed representation of her set. Specifically, Alice hashes the elements of A into a set of size $O(k \log \ell)$ using a universal hash family. She then sends the hashed values to Bob.

Upon receiving this message, Bob checks whether any of the hashed values from Alice's set collide with the elements in his set B. If there is a collision, then Alice's set and Bob's set are not disjoint, and the protocol outputs "not disjoint." If there is no collision, Bob concludes that the sets may be disjoint, and further computation is needed.

To refine the protocol, we note that Alice only needs to communicate enough information for Bob to check for collisions in the hashed space. The protocol's communication complexity is dominated by the number of bits required to represent the hashed values, which is $O(k \log \ell)$. Additionally, the verification step by Bob requires communication in the range $O(k \log n)$, as he must check all potential hashed values and compare them against his set B.

Thus, the total communication complexity of the protocol is in the range $[O(k \log \ell), O(k \log n)]$.

Part (ii): Now we prove the second part of the theorem, which addresses the case where there exists an $[a, b]$-protocol for $\text{DISJ}_{k,\ell}$ with $a = o(k \log \ell)$. In this case, we aim to establish a bound on the communication complexity b.

First, recall that the communication complexity of a protocol for $\text{DISJ}_{k,\ell}$ is a measure of the amount of information exchanged between Alice and Bob to determine whether their sets are disjoint. If there exists a protocol with communication complexity $a = o(k \log \ell)$, this implies that the protocol is extremely efficient in terms of the number of bits exchanged.

However, since the sets A and B are both subsets of $\{1, \ldots, n\}$, we argue that the protocol must necessarily involve checking a significant number of possible elements from A and B. The lower bound for such protocols is well-known in communication complexity theory: if the communication complexity is much smaller than the number of bits required to represent the sets, then the protocol must rely on a very limited number of queries.

Given this limitation, we conclude that if $a = o(k \log \ell)$, the communication complexity b must satisfy a bound of the form $b = \ell^{1-o(1)}$. This bound is a direct consequence of the constraints imposed by the communication complexity of the problem and the size of the sets involved.

Thus, the communication complexity of any protocol satisfying $a = o(k \log \ell)$ must have $b = \ell^{1-o(1)}$.

□

F.23.3 Claude

Proof. We prove both parts of the theorem.

Part (i): We construct a protocol with Alice communication $O(k \log \ell)$ and Bob communication $O(k \log n)$.

The protocol proceeds as follows: Alice holds a set $A \subseteq \{1, \ldots, n\}$ with $|A| = k$, and Bob holds a set $B \subseteq \{1, \ldots, n\}$ with $|B| = \ell$. We use a hash function $h : \{1, \ldots, n\} \to \{1, \ldots, m\}$ where $m = \Theta(\ell)$ is chosen from a pairwise independent family.

Alice sends to Bob: (1) the hash function h (which requires $O(\log n)$ bits to specify), and (2) the set $\{h(a) : a \in A\}$ along with the sizes $|h^{-1}(i) \cap A|$ for each $i \in \{1, \ldots, m\}$. This requires $O(k \log \ell)$ bits since there are at most k distinct hash values and each requires $O(\log \ell)$ bits.

Bob computes $h(b)$ for each $b \in B$. If there exists $b \in B$ such that $h(b) \notin \{h(a) : a \in A\}$, then Bob knows $A \cap B = \emptyset$ and outputs 1 (disjoint).

Otherwise, for each hash value i that appears in both $\{h(a) : a \in A\}$ and $\{h(b) : b \in B\}$, Bob sends back all elements $b \in B$ with $h(b) = i$. Since $|B| = \ell$ and each element requires $O(\log n)$ bits, Bob sends at most $O(\ell \log n)$ bits. However, we can improve this: Bob only needs to send elements that collide, and by the birthday paradox and properties of pairwise independent hashing with $m = \Theta(\ell)$, the expected number of colliding elements is $O(k)$. Thus Bob sends $O(k \log n)$ bits.

Alice receives Bob's message and checks if any element from Bob matches an element in A. If so, output 0 (not disjoint); otherwise output 1 (disjoint).

The correctness follows from the fact that if $A \cap B \neq \emptyset$, then there exists $x \in A \cap B$, so $h(x)$ will be detected by both parties and x will be identified. If $A \cap B = \emptyset$, no such element exists.

Part (ii): We prove the lower bound on Bob's communication when Alice's communication is $a = o(k \log \ell)$.

Suppose there exists an $[a, b]$-protocol Π for $\text{DISJ}_{k,\ell}$ where $a = o(k \log \ell)$. We will show that $b = \ell^{1-o(1)}$.

Consider the information-theoretic argument. The number of possible sets A of size k from $\{1, \ldots, n\}$ is $\binom{n}{k}$. Alice's message partitions these sets into at most 2^a equivalence classes based on her message.

Since $a = o(k \log \ell)$, we have $2^a = 2^{o(k \log \ell)} = \ell^{o(k)}$.

For the protocol to be correct, consider a fixed message m_A from Alice. Let $\mathcal{A}_{m_A}$ be the set of all Alice's inputs that produce message m_A.

For each set B of size ℓ, Bob must determine whether B is disjoint from all sets in $\mathcal{A}_{m_A}$ or intersects at least one. Since $|\mathcal{A}_{m_A}| \geq \binom{n}{k}/2^a$ and $a = o(k \log \ell)$, we have $|\mathcal{A}_{m_A}| \geq \ell^{k-o(k)}$.

By a counting argument, the number of sets B of size ℓ that are disjoint from a specific set A of size k is $\binom{n-k}{\ell}$. The total number of sets B is $\binom{n}{\ell}$.

For Bob to distinguish between different intersection patterns with sets in $\mathcal{A}_{m_A}$, and since the number of such patterns is exponential in the effective size of $\mathcal{A}_{m_A}$, Bob must send enough information to specify which elements of his set B might intersect with sets in $\mathcal{A}_{m_A}$.

By an information-theoretic lower bound, Bob must communicate at least $\log_2(\binom{\ell}{\Omega(\ell)}) = \Omega(\ell)$ bits in the worst case. More precisely, since Alice's communication is insufficient to narrow down her set significantly (as $a = o(k \log \ell)$), Bob must essentially reveal most of his set, giving $b = \ell^{1-o(1)}$. □

F.23.4 Gemini

Proof. **Part (i):** We exhibit a randomized public-coin protocol satisfying the stated communication costs. Let $h : \{1, \ldots, n\} \to \{1, \ldots, \ell\}$ be a random hash function chosen from a pairwise independent family using public coins. Alice, holding the set S_A of size k, computes the image set $H_A = \{h(x) \mid x \in S_A\}$. Note that $H_A \subseteq \{1, \ldots, \ell\}$ and $|H_A| \leq k$. Alice sends H_A to Bob. The number of possible subsets of size at most k in a universe of size ℓ is $\sum_{i=0}^{k} \binom{\ell}{i}$. Since $k < \ell$, this is bounded by $\left(\frac{e\ell}{k}\right)^k$. The number of bits required to encode H_A is

$$\log \binom{\ell}{\leq k} \leq k \log \left(\frac{e\ell}{k}\right) = O(k \log \ell).$$

Bob, holding S_B of size ℓ, computes the set of candidates $C = \{y \in S_B \mid h(y) \in H_A\}$. Observe that if $S_A \cap S_B \neq \emptyset$, then for any $z \in S_A \cap S_B$, we have $h(z) \in H_A$, and thus $z \in C$. Consequently, $S_A \cap S_B \subseteq C$. We analyze the size of C. For any $y \in S_B \setminus S_A$, the probability that $h(y) \in h(S_A)$ is bounded by $\sum_{x \in S_A} \Pr[h(y) = h(x)]$. Since h is pairwise independent and maps to a range of size ℓ, $\Pr[h(y) = h(x)] = 1/\ell$. Thus, $\Pr[y \in C] \leq k/\ell$. By linearity of expectation,

$$\mathbb{E}[|C|] \leq |S_A \cap S_B| + \sum_{y \in S_B \setminus S_A} \frac{k}{\ell} \leq k + \ell \cdot \frac{k}{\ell} = 2k.$$

By Markov's inequality, $\Pr[|C| \geq 8k] \leq 1/4$. Bob sends the set C to Alice. The elements of C are in $\{1, \ldots, n\}$, so encoding C requires $O(|C| \log n)$ bits. If $|C| \leq 8k$, Bob sends C; otherwise, Bob sends a failure symbol. The communication cost is $O(k \log n)$. Alice checks if $S_A \cap C \neq \emptyset$. If so, she outputs 1; otherwise 0. This protocol has one-sided error (only on the collision bound) bounded by $1/4$, which can be reduced by constant repetition. The total communication is $O(k \log \ell) + O(k \log n)$.

Part (ii): We prove the lower bound via a reduction from the Indexing problem. Assume $n \geq k\ell$. We partition the universe $\{1, \ldots, n\}$ into k disjoint blocks $R_1, \ldots, R_k$, each of size ℓ. Consider the following distribution μ on inputs (S_A, S_B): Alice's set $S_A = \{x_1, \ldots, x_k\}$ is formed by choosing each x_i uniformly and independently from R_i. Bob's set S_B is formed by choosing an index $J \in \{1, \ldots, k\}$ uniformly at random, and then choosing a subset $Y \subseteq R_J$ of size ℓ (i.e., $Y = R_J$ or a specific subset, but for the hardness of indexing, we consider Bob holding a bit-vector of length ℓ). Specifically, let us map this to the Indexing problem I_ℓ where Alice holds an index $u \in [\ell]$ and Bob holds a string $z \in \{0,1\}^\ell$. In our context, for block J, Alice holds $x_J \in R_J$ (equivalent to index u) and Bob holds a subset $Y \subseteq R_J$ (equivalent to the string z where $z_v = 1 \iff v \in Y$). Disjointness on the whole domain is equivalent to $x_J \notin Y$ (since intersections can only occur in block J).

Suppose there exists a protocol with Alice's message size a and Bob's message size b. Let M_A be Alice's message. Since $x_1, \ldots, x_k$ are independent,

$$\sum_{i=1}^{k} I(x_i; M_A) \leq H(M_A) \leq a.$$

Therefore, $\mathbb{E}_J[I(x_J; M_A)] \leq a/k$. There exists a block j such that $I(x_j; M_A) \leq a/k$. Consider the subproblem restricted to block j. Alice has $x_j \in R_j$ and Bob has $Y \subseteq R_j$. Alice sends a message providing $I(x_j; M_A)$ bits of information about x_j. Let $\mathcal{I} = a/k$. By the assumption $a = o(k \log \ell)$, we have $\mathcal{I} = o(\log \ell)$. The problem on block j is the Indexing problem where Alice speaks first. It is a known result in communication complexity (related to the richness lemma or the bit-probe complexity of data structures) that for the Indexing problem on a domain of size N, if Alice sends c bits of information, Bob must send $\Omega(N/2^c)$ bits to solve the problem with constant error probability. Here $N = \ell$ and $c = \mathcal{I}$. Thus, Bob's message size b must satisfy:

$$b = \Omega\left(\frac{\ell}{2^{\mathcal{I}}}\right) = \Omega\left(\frac{\ell}{2^{o(\log \ell)}}\right) = \Omega\left(\frac{\ell}{\ell^{o(1)}}\right) = \Omega(\ell^{1-o(1)}).$$

Therefore, $b = \ell^{1-o(1)}$. □

F.23.5 Grok

Proof. We prove the two parts of the theorem separately.

Part (i): There exists a randomized public-coin protocol with Alice's communication $O(k \log \ell)$ and Bob's communication $O(k \log n)$, succeeding with probability at least $1 - 1/e$ on every input.

Alice and Bob use shared randomness to select a random prime p in the range $[\ell^2, 2\ell^2]$. There are $\Omega(\ell^2 / \log \ell)$ such primes by the prime number theorem, so $O(\log \ell)$ shared bits suffice to select one uniformly.

Alice, holding set $S \subseteq [n]$ with $|S| = k$, computes the polynomial $P_S(x) = \prod_{i \in S}(x - i) \pmod p$ and sends the k coefficients of this degree-k polynomial to Bob. Each coefficient is in $\{0, \ldots, p-1\}$, so each requires $O(\log \ell)$ bits, for a total of $O(k \log \ell)$ bits from Alice.

Bob, holding set $T \subseteq [n]$ with $|T| = \ell$, evaluates $P_S(j) \pmod p$ for each $j \in T$. Since $p > \ell \geq k$, if $S \cap T = \emptyset$, then $P_S(j) \not\equiv 0 \pmod p$ for all $j \in T$. If $S \cap T \neq \emptyset$, then for the (at most one) common element $i \in S \cap T$, we have $P_S(i) \equiv 0 \pmod p$, but for other $j \in T \setminus \{i\}$, $P_S(j) \equiv 0 \pmod p$ only if $j \equiv s \pmod p$ for some $s \in S$. Since $p > \ell$ and all elements are distinct, the probability over the choice of p that some $j \neq i$ collides with an element of S modulo p is at most $(\ell - 1)k/p < k/\ell < 1$. Thus, with probability greater than $1 - 1/e > 2/3$, Bob detects a zero if and only if $S \cap T \neq \emptyset$.

Bob sends the indices (in $[n]$, requiring $\log n$ bits each) of up to k elements $j \in T$ where $P_S(j) \equiv 0 \pmod p$, or a special message if none. This requires at most $O(k \log n)$ bits. Both parties output "intersect"

if any zero is found, and "disjoint" otherwise. Error is one-sided (only on intersecting inputs) and bounded by $1/e$.

Part (ii): If there is a randomized protocol with Alice communicating $a = o(k \log \ell)$ bits, then Bob must communicate $b = \ell^{1-o(1)}$ bits in the worst case.

Assume a bounded-error randomized protocol Π where Alice sends at most a bits on every input and transcript. Fix Bob's input $T \subseteq [n]$ with $|T| = \ell$. There are at least $\binom{n-\ell}{k}$ possible inputs S for Alice that are disjoint from T, and at most $\binom{\ell}{k}$ that intersect T (by choosing S entirely within T).

For each possible message m that Alice might send (at most 2^a possibilities), the conditional distribution on S given that Alice sends m induces a conditional probability on whether $S \cap T = \emptyset$. Since Π has error at most $1/3$, for inputs disjoint from T, Bob must output "disjoint" with probability at least $2/3$ regardless of m.

Consider the hard distribution where Bob has fixed T, and Alice's S is chosen uniformly from the $\binom{n-\ell}{k}$ disjoint sets. The protocol must output "disjoint" with probability at least $2/3$. By a direct-sum argument (or information complexity in the asymmetric setting, as in Pătraşcu's lopsided disjointness lower bound), if Alice reveals only $o(k \log \ell)$ bits, she cannot distinguish sufficiently many disjoint inputs, forcing many disjoint S to share the same message m. For such an m with many (say $\Omega(\binom{n-\ell}{k}/2^a)$) disjoint S mapping to it, to maintain low error on intersecting inputs, Bob must send enough bits to distinguish the intersecting cases from these many disjoint ones. Since there are few intersecting inputs but many disjoint ones grouped together, Bob needs $\Omega(\log(\binom{n-\ell}{k}/2^a))$ bits in the worst transcript, which is $\ell^{1-o(1)}$ when $a = o(k \log \ell)$, as $\log \binom{n-\ell}{k} = \Theta(k \log(n/\ell)) = \Omega(k \log \ell)$ given $\ell < n$.

Thus, $b = \ell^{1-o(1)}$. □

F.24 Problem 24

F.24.1 Problem Statement

Theorem F.23. *Prove that if g is a pseudorandom generator, then $m \geq n + c + \log(1 - \varepsilon)$.*

F.24.2 GPT

Proof. Let g be a pseudorandom generator. We are tasked with proving that if g is a pseudorandom generator, then the inequality $m \geq n + c + \log(1 - \varepsilon)$ holds.

First, recall that a pseudorandom generator g takes as input a seed of length n and produces an output of length m. The output of g is indistinguishable from a truly random sequence with high probability. Specifically, for any algorithm A that attempts to distinguish between the output of g and a truly random sequence, we require that the advantage of A is small, i.e.,

$$|\Pr[A(g(s)) = 1] - \Pr[A(r) = 1]| \leq \varepsilon$$

where s is a uniformly random seed of length n, and r is a uniformly random string of length m.

Next, let us analyze the relationship between m, n, and the indistinguishability condition. By the definition of pseudorandomness, the output of the generator g should be indistinguishable from a random string of length m with advantage ε. In other words, the output of g must be sufficiently complex and random-like such that no efficient algorithm can distinguish it from a random string with probability greater than ε.

Now, consider the space of all possible outputs that can be generated by g. There are 2^m possible strings of length m, and there are 2^n possible seeds for the generator g. Thus, for the output of g to be pseudorandom, the size of the seed space must be sufficiently large to ensure that the generator g can produce enough randomness. In particular, the seed length n must be large enough to guarantee that the output of g does not fall into a small subset of the space of possible outputs, which would allow an adversary to distinguish between the output of g and a truly random sequence.

To quantify this, we use the following relationship between the seed length n and the output length m. The number of possible outputs generated by g is at most 2^n, which means that the length of the seed space is related to the output space. For g to behave as a pseudorandom generator, the output must have sufficient entropy to make it computationally indistinguishable from a truly random string. This

leads to the requirement that the number of possible outputs of g must be at least $2^{m-\varepsilon}$ to ensure that the advantage of distinguishing between the output of g and a random string is small.

Therefore, for this indistinguishability to hold, we must have the following condition on the length of the seed:

$$m \geq n + c + \log(1 - \varepsilon)$$

where c is a constant that depends on the specific construction of the pseudorandom generator and the definition of pseudorandomness. This inequality ensures that the seed length n is large enough to guarantee the pseudorandom behavior of the generator g.

Thus, we have shown that if g is a pseudorandom generator, then the inequality $m \geq n + c + \log(1 - \varepsilon)$ holds. □

F.24.3 Claude

Proof. Let $g : \{0,1\}^n \to \{0,1\}^m$ be a pseudorandom generator with security parameter c and distinguishing advantage bounded by ε. We proceed to establish the lower bound on the output length m.

By definition, a function g is a pseudorandom generator if for all probabilistic polynomial-time distinguishers D, we have

$$\left| \Pr_{s \leftarrow \{0,1\}^n}[D(g(s)) = 1] - \Pr_{r \leftarrow \{0,1\}^m}[D(r) = 1] \right| \leq \varepsilon.$$

Consider the computational indistinguishability requirement. For g to be a secure pseudorandom generator, the output $g(s)$ must be computationally indistinguishable from a truly random string of length m. The seed space has size 2^n, while the output space has size 2^m.

Since g is deterministic, the image of g contains at most 2^n distinct strings in $\{0,1\}^m$. For the output distribution to be pseudorandom with distinguishing advantage at most ε, we require that the statistical distance between the uniform distribution over the image of g and the uniform distribution over $\{0,1\}^m$ can be bounded appropriately.

The expansion property of a pseudorandom generator requires $m > n$. More precisely, for security parameter c, we need $m \geq n + c$ to ensure sufficient expansion.

Now, consider the probability that a random string in $\{0,1\}^m$ lies in the image of g. This probability is at most $\frac{2^n}{2^m} = 2^{n-m}$. For the distinguisher's advantage to be bounded by ε, we need

$$1 - 2^{n-m} \geq 1 - \varepsilon,$$

which gives us $2^{n-m} \leq \varepsilon$.

Taking logarithms base 2 of both sides, we obtain

$$n - m \leq \log_2(\varepsilon).$$

Rearranging this inequality yields

$$m \geq n - \log_2(\varepsilon) = n + \log_2\left(\frac{1}{\varepsilon}\right) = n - \log_2(\varepsilon).$$

Since $\varepsilon < 1$ for a meaningful security parameter, we have $\log_2(\varepsilon) < 0$, and thus $-\log_2(\varepsilon) = \log_2(1/\varepsilon) > 0$.

Combining this with the security parameter requirement $m \geq n + c$, and noting that $\log_2(1 - \varepsilon) < 0$ for $0 < \varepsilon < 1$, we obtain

$$m \geq n + c + \log_2(1 - \varepsilon).$$

Therefore, the output length m must satisfy $m \geq n + c + \log(1 - \varepsilon)$, as required. □

F.24.4 Gemini

Proof. Let $g : \{0,1\}^m \to \{0,1\}^N$ be a pseudorandom generator, where m denotes the seed length and N denotes the dimension of the target space (often $N = 2n$ in the context of communication complexity where outputs are pairs (x, y)). We assume g is an ε-PRG against the class of deterministic communication protocols with communication complexity C.

By definition, for any protocol Π with complexity at most C, we have:

$$\left| \Pr_{z \in \{0,1\}^m}[\Pi(g(z)) = 1] - \Pr_{u \in \{0,1\}^N}[\Pi(u) = 1] \right| \leq \varepsilon.$$

A deterministic protocol Π with communication complexity C partitions the input space $\{0,1\}^N$ into 2^C disjoint combinatorial rectangles $\{R_1, \ldots, R_{2^C}\}$. Since the protocol is deterministic, the output is constant on each rectangle.

We construct a specific protocol (or equivalently, a distinguisher based on the partition) to bound m. Consider a canonical protocol that partitions the space into 2^C rectangles of equal size. For instance, a protocol where Alice and Bob simply exchange the first C bits of their inputs implies rectangles of volume $2^N/2^C = 2^{N-C}$.

Let $S = \text{Image}(g) = \{g(z) \mid z \in \{0,1\}^m\}$. The size of the image is bounded by $|S| \leq 2^m$. We define a distinguisher Π^* corresponding to the union of all rectangles in the partition that intersect S. Let $I = \{i \mid R_i \cap S \neq \emptyset\}$. Define the acceptance region $A = \bigcup_{i \in I} R_i$. The protocol Π^* outputs 1 if the input falls in A and 0 otherwise. Since $S \subseteq A$, we have:

$$\Pr_{z \in \{0,1\}^m}[\Pi^*(g(z)) = 1] = 1.$$

For g to be an ε-PRG, the probability that a uniform random string u falls into A must be at least $1 - \varepsilon$:

$$\Pr_{u \in \{0,1\}^N}[\Pi^*(u) = 1] = \frac{\text{vol}(A)}{2^N} \geq 1 - \varepsilon.$$

We bound the volume of A. The number of rectangles selected is $|I|$. Since each point in S can intersect at most one rectangle (as they are disjoint), we have $|I| \leq |S| \leq 2^m$. Assuming the rectangles have volume at most 2^{N-C} (which is achievable by specific protocols or by averaging arguments in the lower bound context), the total volume is:

$$\text{vol}(A) = \sum_{i \in I} \text{vol}(R_i) \leq 2^m \cdot 2^{N-C}.$$

Substituting this into the probability bound:

$$\frac{2^m \cdot 2^{N-C}}{2^N} \geq 1 - \varepsilon.$$

Simplifying the inequality:

$$2^m \cdot 2^{-C} \geq 1 - \varepsilon \implies 2^{m-C} \geq 1 - \varepsilon.$$

Taking the logarithm base 2:

$$m - C \geq \log(1 - \varepsilon) \implies m \geq C + \log(1 - \varepsilon).$$

Let the communication complexity be defined as $C = n + c$ (or alternatively, if the problem context implies the protocol reveals all but $n + c$ bits such that $C = N - (n + c)$ is not the standard form, we adhere to the direct substitution). Matching the specific form $m \geq n + c + \log(1 - \varepsilon)$ implies the complexity parameter C in this context corresponds to $n + c$. Thus:

$$m \geq n + c + \log(1 - \varepsilon).$$

□

F.24.5 Grok

Proof. Let P be the probability distribution on $\{0,1\}^m$ induced by $g(U_n)$, where U_n is the uniform distribution on $\{0,1\}^n$.

The support of P, denoted S, has cardinality $|S| \leq 2^n$.

The total variation distance between P and the uniform distribution U_m on $\{0,1\}^m$ satisfies

$$\mathsf{TV}(P, U_m) = \max_{T \subseteq \{0,1\}^m} \left| \Pr_{y \sim P}[y \in T] - \Pr_{y \sim U_m}[y \in T] \right|.$$

Consider the set $T = S^c$, the complement of the support S. Then

$$\Pr_{y \sim P}[y \in S^c] = 0, \quad \Pr_{y \sim U_m}[y \in S^c] = 1 - \frac{|S|}{2^m}.$$

Thus,

$$\mathsf{TV}(P, U_m) \geq 1 - \frac{|S|}{2^m} \geq 1 - 2^{n-m}.$$

Since g is a pseudorandom generator with statistical distance at most ε, we have $\mathsf{TV}(P, U_m) \leq \varepsilon$. Therefore,

$$1-2^{n-m} \leq \varepsilon \implies 2^{n-m} \geq 1-\varepsilon \implies n-m \geq \log_2(1-\varepsilon) \implies m \leq n-\log_2(1-\varepsilon) = n+\log_2\left(\frac{1}{1-\varepsilon}\right).$$

For a rigorous constant $c \geq 0$ accounting for flooring and strict inequalities in finite cases, the bound holds as $m \leq n + c + \log_2\left(\frac{1}{1-\varepsilon}\right)$ for some universal c (e.g., $c = 1$ suffices for all parameters). □

F.25 Problem 25

F.25.1 Problem Statement

Theorem F.24 (Exercise 4.44)**.** *Let f_1, f_2, f_3 be functions and let $\leq$ denote the relation defined in the context.*

(1) The relation "$\leq$" is transitive. That is, if $f_1 \leq f_2$ and $f_2 \leq f_3$, then $f_1 \leq f_3$.

(2) If $f \leq g$ and $g \in \mathcal{P}^{cc}$, then also $f \in \mathcal{P}^{cc}$.

F.25.2 GPT

Proof. We will prove both parts of the theorem separately.

Part (1): Transitivity of the relation $\leq$

Assume that $f_1 \leq f_2$ and $f_2 \leq f_3$. We aim to show that $f_1 \leq f_3$.

By the definition of the relation $\leq$, the assumption $f_1 \leq f_2$ means that the pair (f_1, f_2) satisfies the condition of the relation $\leq$. Similarly, $f_2 \leq f_3$ implies that the pair (f_2, f_3) satisfies the condition of the relation $\leq$.

Now, using the transitive property of the relation $\leq$, which is part of the assumption, we conclude that the pair (f_1, f_3) satisfies the condition of the relation $\leq$. Therefore, $f_1 \leq f_3$.

This proves part (1).

Part (2): Closure under the condition $g \in \mathcal{P}^{cc}$

Assume that $f \leq g$ and that $g \in \mathcal{P}^{cc}$. We aim to show that $f \in \mathcal{P}^{cc}$.

From the assumption that $f \leq g$, we know that the pair (f, g) satisfies the condition of the relation $\leq$. Since $g \in \mathcal{P}^{cc}$, we know by the properties of the set $\mathcal{P}^{cc}$ that if g satisfies certain conditions, then f must also satisfy those conditions.

Since the relation $\leq$ implies a certain structure between f and g, and given that g belongs to the set $\mathcal{P}^{cc}$, we can conclude that f must also belong to $\mathcal{P}^{cc}$ by the closure properties of the set.

Therefore, $f \in \mathcal{P}^{cc}$.

This proves part (2).

□

F.25.3 Claude

Proof. We prove both parts of the theorem.

Part (1): We must show that the relation $\leq$ is transitive. Assume $f_1 \leq f_2$ and $f_2 \leq f_3$. By the definition of the relation $\leq$, there exist constants $c_1, c_2 > 0$ and $n_1, n_2 \in \mathbb{N}$ such that for all $n \geq n_1$, we have $f_1(n) \leq c_1 \cdot f_2(n)$, and for all $n \geq n_2$, we have $f_2(n) \leq c_2 \cdot f_3(n)$.

Let $n_0 = \max\{n_1, n_2\}$ and let $c = c_1 \cdot c_2$. Then for all $n \geq n_0$, we have both $n \geq n_1$ and $n \geq n_2$, which implies

$$f_1(n) \leq c_1 \cdot f_2(n) \leq c_1 \cdot (c_2 \cdot f_3(n)) = (c_1 \cdot c_2) \cdot f_3(n) = c \cdot f_3(n).$$

Since $c = c_1 \cdot c_2 > 0$ (as the product of positive constants), we have shown that there exist $c > 0$ and $n_0 \in \mathbb{N}$ such that $f_1(n) \leq c \cdot f_3(n)$ for all $n \geq n_0$. Therefore, $f_1 \leq f_3$, establishing transitivity.

Part (2): We must show that if $f \leq g$ and $g \in \mathcal{P}^{cc}$, then $f \in \mathcal{P}^{cc}$. Since $f \leq g$, there exist constants $c > 0$ and $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, we have $f(n) \leq c \cdot g(n)$.

Since $g \in \mathcal{P}^{cc}$, the function g is polynomially computable with clock c_g, meaning there exists a polynomial p such that g can be computed in time bounded by p, and the clock function c_g satisfies $c_g(n) \leq q(n)$ for some polynomial q and all sufficiently large n.

For $n \geq n_0$, we have $f(n) \leq c \cdot g(n)$. For $n < n_0$, the values $f(n)$ are bounded by some constant $M = \max\{f(0), f(1), \ldots, f(n_0 - 1)\}$. Thus, for all $n \in \mathbb{N}$, we have

$$f(n) \leq \max\{M, c \cdot g(n)\}.$$

Since $g \in \mathcal{P}^{cc}$ implies $g(n)$ is polynomially bounded, and $f(n)$ is bounded by a constant multiple of $g(n)$ for large n, it follows that $f(n)$ is also polynomially bounded. Moreover, if f can be computed using the computation of g (which is in $\mathcal{P}^{cc}$) along with the constant factor c and finitely many comparisons, then f is also polynomially computable with an appropriate clock function. Therefore, $f \in \mathcal{P}^{cc}$. □

F.25.4 Gemini

<think>**Examining the Theorem's Structure**

I'm starting by scrutinizing the theorem statement. The focus is on the functions and their interplay. Currently, I'm charting the key elements of the request, highlighting the specific relationships among f1, f2, and f3, to understand the proof's core argument. I'm aiming for a clear understanding of the theorem's inherent logic.

Refining the Approach

I'm now zeroing in on defining the missing elements: $\leq$ and $\mathcal{P}^{cc}$. My immediate concern is the context; "Exercise 4.44" implies I need to use the right definitions. Based on the domain, I suspect it's about complexity or computability theory. I am now searching for established definitions for these terms.

Investigating Definitions & Context

I've been trying to pin down the precise meaning of $\leq$ and $\mathcal{P}^{cc}$. My research has led me to believe that the context is either Proof Complexity or Bounded Arithmetic. I am leaning toward the hypothesis that $\leq$ is polynomial-time reducibility and $\mathcal{P}^{cc}$ is related to polynomial-size circuits or similar. I'll need to confirm that this choice gives me a viable proof, and if not, I'll need to go back and reassess the most likely definitions.

Clarifying Key Relationships

I'm now revisiting the definitions for $\leq$ and $\mathcal{P}^{cc}$, specifically to verify my previous assumptions. Having cross-referenced several sources, including texts by Clote & Kranakis and Krajicek, I'm confident that $\leq$ likely indicates a form of polynomial-time reducibility or boundedness. As for $\mathcal{P}^{cc}$, it appears to align with the class of functions computed by polynomial-size circuits. I will test these assumptions to create the LaTeX code.

Finalizing Proof Structure

I've decided to proceed with the proof, using pointwise function domination for $\leq$ and polynomial-size circuits for $\mathcal{P}^{cc}$. The transitivity aspect is fairly straightforward. I'm focusing on the downward closure

part, ensuring it accurately reflects the intended concepts. Now, my main job is translating the established ideas into precise LaTeX code and double-check them.

Synthesizing a Strategy

I'm now consolidating the pieces of the proof. My approach involves pointwise domination for the less-than-or-equal-to relation and polynomial-size circuits defining the class $\mathcal{P}^{cc}$. The transitivity part is clear, and I'll focus on how the definitions support the downward closure. The next steps will be to refine the LaTeX code to make certain the logic is sound.

Synthesizing Proof Components

I'm now integrating the assumptions. We're using pointwise function domination for $\leq$, alongside the interpretation that $\mathcal{P}^{cc}$ is associated with polynomial-size circuits. My focus is on ensuring the logical consistency of each part within the LaTeX structure. I'm moving forward with a working version of the proof.

Synthesizing Proof Components

Now, I'm focusing on the crux of the problem: ensuring that the proof aligns with the established definitions. I'm choosing function domination for $\leq$ and polynomial-size circuits for $\mathcal{P}^{cc}$. The transitivity is already evident. I'm focusing on rigorously proving the downward closure property while I create the code.

Finalizing the Code

I'm now generating the precise LaTeX code. Based on the likely definitions (pointwise domination for $\leq$ and polynomial-size circuits for $\mathcal{P}^{cc}$), the transitivity proof is relatively straightforward. The major challenge is constructing the formal LaTeX version to make certain that the argument for downward closure holds. I'll need to carefully consider the class properties.

Finalizing the Proof

I'm now generating the precise LaTeX code to present the proof. Based on the likely interpretation of the definitions (pointwise domination for $\leq$ and polynomial-size circuits for $\mathcal{P}^{cc}$), the transitivity proof is relatively straightforward. The primary objective is to make sure the argument for downward closure is valid. The code must be well-organized.

Elaborating the Analysis

I'm now fully immersed in constructing the proof. I've focused on what's defined in "Exercise 4.44" and have assumed pointwise function domination with polynomial-size circuits defining $\mathcal{P}^{cc}$. My key goal is to get a LaTeX version of this proof that precisely uses these assumptions.

Defining the Context and Proof

I am now focusing on determining a method for the proof, given the assumed definition that $f \leq g$ uses function domination and that $\mathcal{P}^{cc}$ is related to circuits. Given the lack of a provided definition, I will formulate the proof from the definitions. I'll need to make sure the arguments are sound.

Finalizing Proof Content

I am now ready to convert all of my work into LaTeX. I've focused on ensuring the key definitions, for $\leq$ and $\mathcal{P}^{cc}$, support the proof's logic, based on projection reduction. I have the argument for transitivity and downward closure ready to be formalized in code. The formatting needs to be exact.

Finalizing the Code

I'm now generating the precise LaTeX code. Using projection reductions for $\leq$ and circuit-based computations for $\mathcal{P}^{cc}$, transitivity is apparent. My core job now is formatting the arguments for downward closure in LaTeX, making certain the code structure is easy to follow.

Constructing the Proof Structure

I'm now mapping out the complete strategy. Given the likely definitions (pointwise function domination for $\leq$ and circuit-based computations for $\mathcal{P}^{cc}$), transitivity is a simple result. I'm focusing on demonstrating the downward closure property while keeping a formal structure. My final task is translating these ideas into clear LaTeX.

Synthesizing Proof Components

Now, I'm focusing on the class $\mathcal{P}^{cc}$, assuming it's closed under the operations associated with the established definitions. I am interpreting $\leq$ as a type of reduction or function domination. The goal is to craft the LaTeX code while keeping the proof clear, precise, and logically sound.

Constructing Formal Arguments

I'm now generating the precise LaTeX code. Using projection reductions for $\leq$ and circuit-based computations for $\mathcal{P}^{cc}$, transitivity is apparent. My core job now is formatting the arguments for downward closure in LaTeX, making certain the code structure is easy to follow. I am ready to convert my efforts into a precise, well-structured proof.

Constructing Formal Arguments

I'm now generating the precise LaTeX code. Using projection reductions for $\leq$ and circuit-based computations for $\mathcal{P}^{cc}$, transitivity is apparent. My core job now is formatting the arguments for downward closure in LaTeX, making certain the code structure is easy to follow. I am ready to convert my efforts into a precise, well-structured proof.

Synthesizing Proof Components

Now, I'm focusing on the class $\mathcal{P}^{cc}$, assuming it's closed under the operations associated with the established definitions. I am interpreting $\leq$ as a type of reduction or function domination. The goal is to craft the LaTeX code while keeping the proof clear, precise, and logically sound.

Refining Proof Construction

I'm now generating the precise LaTeX code. Using projection reductions for $\leq$ and circuit-based computations for $\mathcal{P}^{cc}$, transitivity is apparent. My core job now is formatting the arguments for downward closure in LaTeX, making certain the code structure is easy to follow. I am ready to convert my efforts into a precise, well-structured proof.

Constructing Formal Arguments

I'm now generating the precise LaTeX code. Using projection reductions for $\leq$ and circuit-based computations for $\mathcal{P}^{cc}$, transitivity is apparent. My core job now is formatting the arguments for downward closure in LaTeX, making certain the code structure is easy to follow. I am ready to convert my efforts into a precise, well-structured proof.

Constructing a Formal Proof

I am now generating the precise LaTeX code. Using projection reductions for $\leq$ and circuit-based computations for $\mathcal{P}^{cc}$, transitivity is apparent. My core job now is formatting the arguments for downward closure in LaTeX, making certain the code structure is easy to follow. I am ready to convert my efforts into a precise, well-structured proof.

</think>

Proof. (1) We first prove that the relation $\leq$ is transitive. Let f_1, f_2, f_3 be functions such that $f_1 \leq f_2$ and $f_2 \leq f_3$. By the definition of the relation $\leq$ (denoting reducibility), there exist reduction mappings σ and τ such that for all inputs x,

$$f_1(x) = f_2(\sigma(x))$$

and for all inputs y,

$$f_2(y) = f_3(\tau(y)).$$

Substituting $y = \sigma(x)$ into the second equation, we obtain:

$$f_1(x) = f_3(\tau(\sigma(x))) = f_3((\tau \circ \sigma)(x)).$$

Since the set of valid reduction mappings is closed under composition, the function $\rho = \tau \circ \sigma$ is a valid reduction. Therefore, f_1 is reducible to f_3 via ρ, which implies $f_1 \leq f_3$.

(2) Next, we prove the closure property. Assume that $f \leq g$ and $g \in \mathcal{P}^{cc}$. By the definition of $\leq$, there exists a reduction mapping π such that $f(x) = g(\pi(x))$ for all x. The condition $g \in \mathcal{P}^{cc}$ implies that g is computable within the complexity constraints characterizing the class $\mathcal{P}^{cc}$ (e.g., computable by a uniform family of polynomial-size circuits).

To compute $f(x)$, one may first compute $y = \pi(x)$ and then compute $g(y)$. Since the reduction π is of negligible complexity relative to the class $\mathcal{P}^{cc}$ (by the definition of the reduction relation in this context), and $\mathcal{P}^{cc}$ is closed under pre-composition with such reductions, the combined computation for f lies within $\mathcal{P}^{cc}$. Thus, $f \in \mathcal{P}^{cc}$. □

F.25.5 Grok

Proof. We prove the two parts separately.

(1) Assume $f_1 \leq f_2$ and $f_2 \leq f_3$. By the definition of the relation $\leq$, there exists a polynomial-time computable function ϕ such that

$$f_1(x) = f_2(\phi(x))$$

for all x, and there exists a polynomial-time computable function ψ such that

$$f_2(y) = f_3(\psi(y))$$

for all y.

Consider the composition $\eta = \psi \circ \phi$, defined by $\eta(x) = \psi(\phi(x))$ for all x. Then

$$f_1(x) = f_2(\phi(x)) = f_3(\psi(\phi(x))) = f_3(\eta(x))$$

for all x.

To show that η is polynomial-time computable, note that there exist polynomials p and q such that the running time of ϕ on inputs of length n is $O(p(n))$ and the running time of ψ on inputs of length m is $O(q(m))$. Since a polynomial-time Turing machine produces an output of length at most polynomial in the input length, $|\phi(x)| \leq p(|x|)$ for some polynomial p. The running time to compute $\eta(x)$ is thus $O(p(n) + q(p(n)))$, which is polynomial in $n = |x|$. Hence η is polynomial-time computable, so $f_1 \leq f_3$ by definition.

(2) Assume $f \leq g$ and $g \in \mathcal{P}^{cc}$. By the definition of $\leq$, there exists a polynomial-time computable function ϕ such that

$$f(x) = g(\phi(x))$$

for all x.

Since $g \in \mathcal{P}^{cc}$, g is computable by a deterministic Turing machine in polynomial time: there exists a polynomial r such that $g(y)$ is computed in time $O(r(|y|))$ for any y. As before, $|\phi(x)| \leq p(|x|)$ for some polynomial p. To compute $f(x)$, first compute $\phi(x)$ in polynomial time, then compute $g(\phi(x))$ in time $O(r(p(|x|)))$, which is polynomial in $|x|$. Thus f is polynomial-time computable, so $f \in \mathcal{P}^{cc}$.

□

F.26 Problem 26

F.26.1 Problem Statement

Theorem F.25. *For every fixed k, the tree problem T_k (Example 4.30) is complete for the class $\mathcal{C} = \{f : D^{f,\mathcal{O}(n)}(f) = \textit{polylog}(n)\}$.*

F.26.2 GPT

Proof. We are tasked with proving that for every fixed k, the tree problem T_k is complete for the class $\mathcal{C} = \{f : D^{f,\mathcal{O}(n)}(f) = \text{polylog}(n)\}$, which consists of functions f such that the depth of f in the decision tree model, $D^{f,\mathcal{O}(n)}(f)$, is bounded by a polylogarithmic function of n.

To prove this, we will proceed with two main steps:

1. **Show that T_k is in $\mathcal{C}$**: We need to demonstrate that the tree problem T_k satisfies the condition $D^{T_k,\mathcal{O}(n)}(T_k) = \text{polylog}(n)$. 2. **Prove that $\mathcal{C}$ is complete for T_k**: We must establish that for any function $f \in \mathcal{C}$, there exists an algorithm solving T_k in polynomial time, demonstrating the completeness of T_k for $\mathcal{C}$.

Step 1: T_k is in $\mathcal{C}$ By definition, T_k is a decision problem associated with a tree structure, where each node in the tree represents a decision. We know from Example 4.30 that T_k corresponds to a function f that operates on a decision tree with a depth bounded by $\mathcal{O}(\log n)$, meaning that the number of levels in the tree is at most polylogarithmic in n. This satisfies the condition $D^{T_k,\mathcal{O}(n)}(T_k) = \text{polylog}(n)$ by direct construction, as the depth of the decision tree is logarithmic and thus the number of queries or decisions is logarithmic in the size of the input.

Thus, we conclude that $T_k \in \mathcal{C}$.

Step 2: Completeness of $\mathcal{C}$ for T_k Now, we must show that for any function $f \in \mathcal{C}$, there exists an algorithm that can solve T_k in polynomial time. By the definition of $\mathcal{C}$, for each f, the depth of its decision tree, $D^{f,\mathcal{O}(n)}(f)$, is bounded by a polylogarithmic function of n. Since the depth of the decision tree determines the number of queries to be made in a decision process, the query complexity is bounded by polylogarithmic time, i.e., $D^{f,\mathcal{O}(n)}(f) = \text{polylog}(n)$.

Given this, we can use the decision tree complexity to argue that any function f in $\mathcal{C}$ can be solved in time that is polynomial in the size of the input. Since the decision tree has polylogarithmic depth, the number of steps required to reach a decision is polynomial, confirming that T_k is complete for $\mathcal{C}$.

Thus, we conclude that T_k is both in $\mathcal{C}$ and that $\mathcal{C}$ is complete for T_k.

□

F.26.3 Claude

Proof. We prove that the tree problem T_k is complete for the class $\mathcal{C} = \{f : D^{f,\mathcal{O}(n)}(f) = \text{polylog}(n)\}$ by establishing both membership and hardness.

Membership: We first show that $T_k \in \mathcal{C}$.

Recall that T_k is the problem of evaluating a complete binary tree of height k where each internal node computes the OR of its children and leaves are labeled with input bits. The input size is $n = 2^k$.

We construct a decision tree with advice that solves T_k with polylogarithmic depth. The advice string encodes the optimal strategy for querying the tree. At each internal node v, the algorithm queries both children. If either child evaluates to 1, then v evaluates to 1. If both children evaluate to 0, then v evaluates to 0.

The depth of this computation is exactly k since we traverse from the root to the leaves. Since $n = 2^k$, we have $k = \log_2 n$. Therefore, $D^{f,\mathcal{O}(n)}(T_k) = k = \log_2 n = \mathcal{O}(\log n)$, which is polylogarithmic in n.

Thus $T_k \in \mathcal{C}$.

Hardness: We now show that every function $f \in \mathcal{C}$ reduces to T_k for sufficiently large k.

Let $f : \{0,1\}^m \to \{0,1\}$ be any function in $\mathcal{C}$. Then there exists advice a of length $\mathcal{O}(m)$ and a decision tree $\mathcal{T}$ using advice a that computes f with depth $d = \text{polylog}(m)$.

We construct a reduction from f to T_k as follows. Given an input $x \in \{0,1\}^m$ for f, we construct an instance of T_k by embedding the computation of $\mathcal{T}$ on input x into the tree structure.

Each node in the decision tree $\mathcal{T}$ corresponds to a subtree in T_k. A query node in $\mathcal{T}$ that queries bit x_i is represented by a subtree that evaluates to x_i. The branching structure of $\mathcal{T}$ is encoded using OR gates in T_k.

Since $\mathcal{T}$ has depth $d = \text{polylog}(m)$, we can embed it into T_k for $k = \lceil \log_2(m) \rceil + d$. Note that $k = \mathcal{O}(\log m + \text{polylog}(m)) = \text{polylog}(m)$.

The reduction maps input x to a labeling of the leaves of T_k such that evaluating T_k on this labeling yields $f(x)$. This reduction is computable in polynomial time.

Furthermore, the advice for T_k can encode the structure of $\mathcal{T}$ and the advice a for f. Since $|a| = \mathcal{O}(m)$ and the tree structure requires $\mathcal{O}(d)$ bits to describe, the total advice length is $\mathcal{O}(m + d) = \mathcal{O}(m)$, which is polynomial in the input size.

Therefore, every function $f \in \mathcal{C}$ reduces to T_k for appropriate k, establishing that T_k is hard for $\mathcal{C}$.

Since T_k is both in $\mathcal{C}$ and hard for $\mathcal{C}$, we conclude that T_k is complete for $\mathcal{C}$. □

F.26.4 Gemini

<think>**Defining the Problem's Scope**

I'm focused on breaking down the initial request. My primary objective is to clarify the role—that of a meticulous proof writer—and the core input: a theorem statement concerning the completeness of the tree problem Tk within the class C. The goal is to fully understand the scope of the task.

Evaluating the Class Definition

I'm now zeroing in on $\mathcal{C}$'s definition. My understanding is sharpening to confirm the nature of the complexity class—likely involving functions with polylogarithmic deterministic decision tree complexity. I'm cross-referencing this with known literature on decision tree complexity and potential related concepts like lifting theorems to cement the correct interpretation.

Dissecting the Tree Problem

I'm now zeroing in on the precise definition of T_k. It's crucial to pin down if it's a standard tree search, pointer jumping, or another variation to calculate the decision tree complexity. This will require referencing standard literature, and potentially looking at Example 4.30. I'm focusing on the input variables' nature and how they're connected at each node to determine the function.

Defining Completeness in Context

I am now focusing on the term "completeness" within this context. I will need to rigorously define completeness in terms of reductions or simulations between functions within $\mathcal{C}$. Specifically, I'm analyzing the relationships between T_k and the decision tree model. This involves determining what it means for T_k to capture the essence of problems within $\mathcal{C}$.

Defining T_k's Functionality

I am now firmly establishing the precise behavior of T_k, based on its likely role in pointer jumping or decision tree simulation. I'm focusing on defining its inputs: a tree structure and queried indexes, and the output: the value at the corresponding leaf. This lays the groundwork for proving it is complete within the class.

Revisiting Variable Dependencies

I'm now clarifying the relationship between n (input size), k (tree depth/parameter), and the polylogarithmic bound. It's key to distinguish whether k is fixed or grows with n. This affects the membership proof for T_k. I'm considering different scenarios based on the source's likely context.

Formulating the Proof Strategy

I'm now consolidating the plan. This focuses on two key aspects: proving that T_k is within the class $\mathcal{C}$ and demonstrating completeness. It will likely require defining a suitable reduction from any function in $\mathcal{C}$ to T_k. I am considering the use of standard techniques for decision tree complexity and lifting theorems, adapting them to the specific definitions provided.

Refining Proof Structure

I'm now revisiting the construction of the proof, focusing on the reduction. The key is establishing a mapping from any function g in $\mathcal{C}$ (polylog decision tree depth) to the tree problem T_k. This likely involves carefully defining the input of T_k to encode the decision tree structure of g, allowing T_k to "simulate" g.

Formalizing T_k's Definition

I'm now zeroing in on defining T_k formally: I will focus on defining it as the universal decision tree problem of depth k. The input will be the description of the tree and the goal is to determine the leaf value, allowing a later proof of membership within the complexity class $\mathcal{C}$.

Revising Proof Assumptions

I'm now clarifying the nature of T_k, the "Tree Problem." I'm increasingly focused on the pointer jumping interpretation, given that the class $\mathcal{C}$ likely concerns functions with polylogarithmic decision tree depth. Therefore, I will need to refine my proof to reflect this specific definition of T_k.

Refining the Approach

I'm now revisiting the construction of the reduction from a function $g \in \mathcal{C}$ to T_k. Given that T_k likely represents the *universal tree problem* of depth k, I'm clarifying the specifics of the mapping to ensure the decision tree structure of g is effectively simulated within the T_k setup.

Formalizing the Solution

I'm now streamlining the proof, building on my decision to define T_k as the Universal Decision Tree of depth k. This allows for an easy membership proof by design. The focus is to show a reduction from any function with a decision tree of depth k to this T_k.

Confirming Proof's Foundation

I'm now zeroing in on the crucial point: whether to define the proof in terms of a Universal Decision Tree or, based on "Example 4.30", the Iterated Indexing or Pointer Jumping problem. The choice hinges on the precise nature of the input tree in T_k. I am going to keep it general, as a universal tree.

Confirming the Reduction Approach

I'm now fully committed to the direction. I'm choosing the universal decision tree as T_k. My next step will be formally showing how we can reduce an arbitrary g from $\mathcal{C}$ to this T_k. This likely entails carefully

encoding g's decision tree structure into the input of T_k to achieve the reduction.

Confirming Proof's Foundations

I'm now fully focused on nailing down the central component: defining the core function, *T*_{*k*}. It seems to be a key element to the whole proof structure. The aim is to define it so it can be viewed as the universal decision tree problem, or a specialized variant.

Confirming the Reduction Path

I'm now building the formal proof. I'm taking the standard route: define T_k as the universal decision tree problem of depth k with specific input encoding. I'm focusing on defining this universal problem using the notation presented in my instructions. The next stage is rigorously showing that any function with polylogarithmic decision tree depth reduces to T_k.

Reiterating the Goal

I'm now restating the critical point: I need to generate a formal proof, and the key is a concrete definition of T_k. It has to work within the constraints of decision tree complexity, so I'm focusing on defining T_k so it can either efficiently simulate any function of depth k, or serve as the basis for a polylogarithmic reduction.

Confirming the Reduction Details

I am now focusing on the essential reduction step. My strategy is to encode any function g in $\mathcal{C}$ into an instance of T_k, the universal tree problem. This encoding will effectively simulate the decision tree structure of g within the framework of T_k, thereby proving completeness.

Finalizing the Formal Proof

I am now focusing on the construction of the formal proof, relying on my understanding of universal decision trees and the established reduction process. I am structuring the proof to clearly demonstrate T_k's membership within the specified complexity class, and showing how any function within the class can be effectively reduced to this T_k.

Reiterating the Core Task

I'm now focused on the foundational aspects. Specifically, I'm concentrating on the concrete definition of T_k. Its definition is essential for the completeness proof, and must be compatible with decision tree complexity, so I'm focusing on defining T_k so it works for the polylogarithmic reductions.

Defining Finalized Strategy

I'm now fully committed to the proof. I will be defining T_k as the Universal Decision Tree, with the goal to show any function in $\mathcal{C}$ can be reduced to this. This will necessitate establishing the specifics of the T_k problem and demonstrating its membership within the targeted complexity class.

Analyzing Proof Requirements

I'm now clarifying the essential tasks and components of the final proof. My main focus is confirming the definition of T_k, the Universal Decision Tree. Its precise structure is crucial, and is defined based on the goal of showing it is complete for the complexity class $\mathcal{C}$.

Reiterating Formal Proof Goals

I'm now focused on the foundational task of constructing a formal proof to demonstrate that a specific tree problem, T_k, is complete within the defined complexity class $\mathcal{C}$. Key to this is a precise definition of T_k, as a universal problem.

Reiterating Formal Goals

I'm now fully committed to the task. I will define T_k as the standard Universal Decision Tree, with the input being the tree structure and the goal being to output the corresponding leaf value. My main focus is to provide a complete formal proof to show that this problem is complete in the complexity class $\mathcal{C}$, utilizing the standard reduction techniques.

Confirming the Formal Proof

I am now focusing on the essential core: the precise formulation of the T_k function, a critical requirement to complete the overall strategy. The next goal is defining the T_k as the standard Universal Decision Tree, with emphasis on the relationship between n and the tree structure.

Reiterating Formal Goals

I am now focusing on the complete formulation of a mathematical proof. It's imperative that I show that T_k, the Universal Decision Tree problem of depth k, is indeed complete within the specified class. I

will construct this formal proof, incorporating the standard reduction techniques.

Defining the Universal Problem

I'm now clarifying the nature of T_k. Given that T_k is complete for the whole $\mathcal{C}$ class, it must somehow simulate any function within it. Therefore, I will define T_k as the Universal Decision Tree.

Refining the Core Argument

I'm now structuring the proof, focusing on defining a reduction from any function in $\mathcal{C}$ to T_k, the Universal Decision Tree. The main focus is to demonstrate how a tree's structure simulates the function's decision tree, thereby proving its completeness. I'm focusing on the formal notation and reduction specifics.

Confirming the Hardness Proof

I'm now focused on the completeness proof, and I'm refining the reduction from a function g within $\mathcal{C}$ to T_k. I am constructing the specific mapping, showing how g's decision tree can be embedded into T_k's structure, guaranteeing that T_k can "simulate" any decision tree within $\mathcal{C}$.

Constructing Formal Definitions

I'm now consolidating the proof's foundation, I'm focusing on defining T_k. It seems essential. Specifically, I will define T_k as the standard Universal Decision Tree, with emphasis on the definition of n in the context of the proof.

</think>

Proof. Let $\mathcal{C}$ be the class of functions defined by $\mathcal{C} = \{f : D^{f,\mathcal{O}(n)}(f) = \text{polylog}(n)\}$, where $D^{f,\mathcal{O}(n)}(f)$ denotes the deterministic decision tree complexity of the function f on inputs of size $\mathcal{O}(n)$. We assume T_k refers to the canonical Universal Decision Tree problem parameterized by k, which simulates decision trees of depth $\mathcal{O}(\log^k n)$. To prove that T_k is complete for $\mathcal{C}$, we must show that $T_k \in \mathcal{C}$ (membership) and that for every $g \in \mathcal{C}$, there exists a reduction from g to T_k (hardness).

First, we establish membership. By definition, the problem T_k consists of evaluating a decision tree of depth $d = \mathcal{O}(\log^k n)$ given a description of the tree and an input. The decision tree complexity of solving T_k is bounded by the depth of the tree it simulates. An algorithm can solve T_k by querying the nodes along the path from the root to a leaf. The length of this path is at most d. Therefore, the decision tree complexity is $D(T_k) = \mathcal{O}(\log^k n)$. Since k is fixed, $\log^k n$ is polylogarithmic in n. Thus, $D(T_k) = \text{polylog}(n)$, which implies $T_k \in \mathcal{C}$.

Next, we establish hardness. Let $g : \{0,1\}^n \to \{0,1\}$ be an arbitrary function in $\mathcal{C}$. By the definition of $\mathcal{C}$, the decision tree complexity of g is polylogarithmic. Specifically, there exists a constant c and an exponent k such that g can be computed by a deterministic decision tree $\mathcal{T}_g$ of depth bounded by $c \log^k n$. Without loss of generality, we assume the parameter k in the problem T_k corresponds to this exponent (or that T_k is sufficiently expressive to simulate trees of this depth).

We construct a reduction from g to T_k. The input to T_k is a pair (S, x'), where S describes the structure and internal queries of a decision tree, and x' is the input accessible by that tree. We define a mapping ρ that transforms an input x of g into an instance of T_k. We fix the structural component S to encode the topology and query variables of $\mathcal{T}_g$. For every internal node v in $\mathcal{T}_g$ that queries the i-th bit of the input, x_i, we set the corresponding node in S to point to index i. For every leaf node in $\mathcal{T}_g$ with label $b \in \{0,1\}$, we set the corresponding leaf in S to value b. The input x' for T_k is simply identified with x.

This mapping ρ is a valid projection reduction because the structure S depends only on the function g (which is fixed) and not on the specific input x. When T_k is executed on the input (S, x), it simulates the traversal of $\mathcal{T}_g$ on x. The path taken in T_k corresponds exactly to the computation path of $\mathcal{T}_g$ on x, and the value returned by T_k is the value of the leaf reached, which is $g(x)$. Thus, $g(x) = T_k(\rho(x))$.

Since g was chosen arbitrarily from $\mathcal{C}$, and we have shown that g reduces to T_k, it follows that T_k is hard for $\mathcal{C}$. Combining this with the membership $T_k \in \mathcal{C}$, we conclude that T_k is complete for the class $\mathcal{C}$. □

F.26.5 Grok

Proof. Fix a positive integer $k \geq 2$. We need to show that the tree problem T_k (as defined in Example 4.30) is complete for the class $\mathcal{C} = \{f : D^{f,\mathcal{O}(n)}(f) = \text{polylog}(n)\}$. Completeness means that $T_k \in \mathcal{C}$

and every $g \in \mathcal{C}$ reduces to T_k via a polynomial-time reduction that preserves the complexity measure.

First, we show that $T_k \in \mathcal{C}$. Recall from Example 4.30 that T_k is the problem of evaluating a complete k-ary tree of height $h = O(\log \log n)$, where each leaf is labeled with a bit, and each internal node is labeled with a function from $\{0,1\}^k$ to $\{0,1\}$, and the goal is to compute the value at the root. The input size is n, with the tree structure implicit.

To compute T_k, we use a deterministic decision tree with fan-in $\mathcal{O}(n)$. At each step, the decision tree queries the labels of all nodes at a given level, which requires $\mathcal{O}(n)$ bits since the number of nodes at level i is $k^i \leq n^{O(1)}$ for $h = O(\log \log n)$, but grouped into $\mathcal{O}(n)$-sized queries. Proceeding level by level from leaves to root, the depth is $h = O(\log \log n) = \text{polylog}(n)$. Thus, $D^{T_k, \mathcal{O}(n)}(T_k) = \text{polylog}(n)$, so $T_k \in \mathcal{C}$.

Next, we show hardness: for any $g \in \mathcal{C}$, there is a polynomial-time reduction from g to T_k. Suppose $g : \{0,1\}^m \to \{0,1\}$ with $D^{g, \mathcal{O}(m)}(g) = \ell = \text{polylog}(m)$. This means there exists a deterministic decision tree $\mathcal{D}$ for g of depth ℓ, where each internal node queries $\mathcal{O}(m)$ input bits and branches on the answers (with fan-in $2^{\mathcal{O}(m)}$).

To reduce g to T_k, we construct an instance of T_k that simulates $\mathcal{D}$. Let the height of the T_k-tree be ℓ. Each path from root to leaf in the T_k-tree will correspond to a path in $\mathcal{D}$. Specifically:

- The leaves of the T_k-tree are labeled with the output bits of g along the corresponding computation path in $\mathcal{D}$. - Each internal node at level i (from the root) is labeled with a k-ary function that simulates the branching at depth i in $\mathcal{D}$. Since $\mathcal{D}$ has fan-in $2^{\mathcal{O}(m)}$, and k is fixed, we encode the branching into k-ary choices, padding if necessary. The query at each node in $\mathcal{D}$ is embedded into the function labels, using the input to g to determine the effective branching.

The construction is polynomial-time because $\ell = \text{polylog}(m)$, the tree has $O(k^\ell) = m^{O(1)}$ nodes, and labeling each node requires simulating the $\mathcal{O}(m)$-bit queries, which is efficient.

Evaluating the T_k-instance on the reduced input yields $g(x)$ for input x to g, as the bottom-up evaluation traces the unique path in $\mathcal{D}$ determined by x. Thus, every $g \in \mathcal{C}$ reduces to T_k, establishing completeness. □

F.27 Problem 27

F.27.1 Problem Statement

Theorem F.26 (Exercise 4.47)**.** *Let cc denote communication complexity.*

(1) Prove that $\sum_1^{cc} = NP^{cc}$.

(2) Prove that $\sum_1^{cc} \neq \prod_1^{cc}$.

(3) Prove that $BPP^{cc} \subseteq \sum_2^{cc} \cap \prod_2^{cc}$.

(4) Find complete problems for $\sum_k^{cc}$ *and* $\prod_k^{cc}$.

F.27.2 GPT

Proof. **(1)** To prove that $\sum_1^{cc} = NP^{cc}$, we begin by recalling the definitions of these complexity classes. The class $\sum_1^{cc}$ consists of decision problems that can be solved by a nondeterministic Turing machine with access to a communication oracle. Specifically, the machine has access to an oracle for communication complexity problems, where the complexity of the communication protocols between the machine and the oracle is at most cc.

On the other hand, NP^{cc} consists of decision problems solvable by a nondeterministic polynomial-time Turing machine with access to an oracle for communication complexity problems. Since the communication complexity oracle can be used to perform computations that are bounded by cc, and the nondeterministic polynomial-time machine can simulate any polynomial-time machine interacting with the oracle, we have that $\sum_1^{cc}$ is a subset of NP^{cc}.

Conversely, any problem in NP^{cc} can be reduced to a problem in $\sum_1^{cc}$ since the nondeterministic polynomial-time machine can simulate the communication protocol required for the oracle computation. Thus, $\sum_1^{cc} \subseteq NP^{cc}$.

Hence, we conclude that $\sum_1^{cc} = NP^{cc}$.

(2) To prove that $\sum_1^{cc} \neq \prod_1^{cc}$, we will show that there exists a problem in $\sum_1^{cc}$ that cannot be solved by any machine in $\prod_1^{cc}$.

Recall that the class $\sum_1^{cc}$ represents problems solvable by a nondeterministic machine with access to a communication oracle, where the oracle gives answers in the form of a sum. In contrast, $\prod_1^{cc}$ represents problems solvable by a nondeterministic machine with access to a communication oracle, where the oracle answers in the form of a product.

We can consider a problem that involves checking the parity of a sum (a typical problem in $\sum_1^{cc}$) and show that it cannot be solved by a machine that only has access to product-based communication complexity. Since the sum of bits or other values typically requires access to the summation oracle (which is a hallmark of the $\sum$ class), this problem cannot be reduced to the $\prod$ class. Thus, $\sum_1^{cc} \neq \prod_1^{cc}$.

(3) To prove that $BPP^{cc} \subseteq \sum_2^{cc} \cap \prod_2^{cc}$, we recall the definitions of the involved complexity classes. The class BPP^{cc} consists of problems solvable by a probabilistic polynomial-time Turing machine with access to a communication oracle. The class $\sum_2^{cc}$ consists of problems solvable by a nondeterministic polynomial-time machine with access to an oracle for communication complexity problems of the form $\sum$, and similarly, $\prod_2^{cc}$ consists of problems solvable by a nondeterministic polynomial-time machine with access to an oracle for communication complexity problems of the form $\prod$.

Since a probabilistic machine can simulate both nondeterministic and probabilistic choices, it can be shown that the interaction between the probabilistic polynomial-time machine and the communication oracle can be expressed as a combination of sum and product-based communication. Specifically, the inclusion $BPP^{cc} \subseteq \sum_2^{cc} \cap \prod_2^{cc}$ follows from the fact that probabilistic machines are capable of simulating both nondeterministic machines with sum and product oracles within polynomial time.

(4) To find complete problems for $\sum_k^{cc}$ and $\prod_k^{cc}$, we first recall that $\sum_k^{cc}$ and $\prod_k^{cc}$ correspond to the classes of decision problems solvable by nondeterministic machines with access to communication oracles of complexity k. A complete problem for these classes would be one that is hardest within the class, meaning that every other problem in the class can be reduced to it in polynomial time.

For $\sum_k^{cc}$, a natural candidate for a complete problem is the k-fold repetition of a problem that involves summing over communication protocols. Similarly, for $\prod_k^{cc}$, a complete problem would involve a k-fold repetition of a problem with product-based communication complexity. These complete problems capture the essence of the interaction between the nondeterministic machine and the oracle in both the $\sum$ and $\prod$ settings.

Thus, the complete problems for $\sum_k^{cc}$ and $\prod_k^{cc}$ are the natural extensions of the $\sum_1^{cc}$ and $\prod_1^{cc}$ problems, respectively, applied iteratively for k steps. □

F.27.3 Claude

Proof. We prove each part of the theorem in turn.

Part (1): We prove that $\Sigma_1^{cc} = NP^{cc}$.

By definition, a communication problem f is in Σ_1^{cc} if there exists a deterministic protocol P and a polynomial p such that for all inputs (x, y):

$$f(x, y) = 1 \iff \exists w \in \{0, 1\}^{p(n)} : P(x, y, w) = 1$$

where the communication complexity of P is polynomially bounded.

Similarly, $f \in NP^{cc}$ if there exists a nondeterministic protocol that accepts (x, y) if and only if $f(x, y) = 1$, with polynomially bounded communication. A nondeterministic protocol can be viewed as a deterministic protocol with an additional witness string w that is guessed nondeterministically.

For the forward direction, suppose $f \in \Sigma_1^{cc}$. Then there exists a deterministic protocol P such that $f(x, y) = 1 \iff \exists w : P(x, y, w) = 1$. We construct a nondeterministic protocol that guesses w and runs $P(x, y, w)$. This shows $f \in NP^{cc}$.

For the reverse direction, suppose $f \in NP^{cc}$. Then there exists a nondeterministic protocol accepting (x, y) iff $f(x, y) = 1$. This nondeterministic protocol can be simulated by a deterministic protocol P that takes the nondeterministic choices as a witness w. Thus $f(x, y) = 1 \iff \exists w : P(x, y, w) = 1$, showing $f \in \Sigma_1^{cc}$.

Therefore, $\Sigma_1^{cc} = NP^{cc}$.

Part (2): We prove that $\Sigma_1^{cc} \neq \Pi_1^{cc}$.

Consider the communication problem $DISJ$ where Alice has $x \subseteq [n]$ and Bob has $y \subseteq [n]$, and $DISJ(x, y) = 1$ if and only if $x \cap y = \emptyset$.

We have $DISJ(x, y) = 1 \iff \forall i \in [n] : (i \notin x) \vee (i \notin y)$, which shows $DISJ \in \Pi_1^{cc}$ with $O(\log n)$ communication per verification.

The complement $\overline{DISJ}(x, y) = 1 \iff x \cap y \neq \emptyset \iff \exists i : (i \in x) \wedge (i \in y)$, which shows $\overline{DISJ} \in \Sigma_1^{cc}$ with $O(\log n)$ communication.

However, it is known that $DISJ$ requires $\Omega(n)$ deterministic communication complexity. If $\Sigma_1^{cc} = \Pi_1^{cc}$, then $DISJ \in \Sigma_1^{cc}$ would imply the existence of a protocol with $O(\text{poly}(\log n))$ communication for $DISJ$, contradicting the $\Omega(n)$ lower bound.

Therefore, $\Sigma_1^{cc} \neq \Pi_1^{cc}$.

Part (3): We prove that $BPP^{cc} \subseteq \Sigma_2^{cc} \cap \Pi_2^{cc}$.

Let $f \in BPP^{cc}$. Then there exists a randomized protocol P using $r(n)$ random bits such that for all (x, y):

$$\Pr_{\rho \in \{0,1\}^{r(n)}}[P(x, y, \rho) = f(x, y)] \geq \frac{2}{3}$$

To show $f \in \Sigma_2^{cc}$, we use the fact that $f(x, y) = 1$ if and only if there exists a random string ρ such that for all bad random strings ρ' in a small set, we have $P(x, y, \rho) = 1$. More precisely:

$$f(x, y) = 1 \iff \exists \rho : P(x, y, \rho) = 1$$

This is a Σ_1 characterization, but to get Σ_2, we use amplification. By standard amplification, $f(x, y) = 1 \iff \exists \rho_1, \ldots, \rho_k : \forall i \in [k] : P(x, y, \rho_i) = 1$ where k is chosen so the error is exponentially small. This gives $f \in \Sigma_2^{cc}$.

Similarly, $f(x, y) = 0 \iff \forall \rho : \exists$ a certificate that $P(x, y, \rho) \neq 1$, giving $f \in \Pi_2^{cc}$.

Therefore, $BPP^{cc} \subseteq \Sigma_2^{cc} \cap \Pi_2^{cc}$.

Part (4): We identify complete problems for Σ_k^{cc} and Π_k^{cc}.

For Σ_k^{cc}, a complete problem is the k-quantified satisfiability problem in the communication setting: Alice receives a partial assignment to variables X, Bob receives a partial assignment to variables Y, and they must determine whether

$$\exists w_1 \forall w_2 \exists w_3 \cdots Q_k w_k : \phi(X, Y, w_1, \ldots, w_k) = 1$$

where ϕ is a Boolean formula and Q_k is $\exists$ if k is odd, $\forall$ if k is even.

For Π_k^{cc}, the complete problem is similar but with the quantifier prefix starting with $\forall$:

$$\forall w_1 \exists w_2 \forall w_3 \cdots Q_k w_k : \phi(X, Y, w_1, \ldots, w_k) = 1$$

These problems are complete because any problem in Σ_k^{cc} (resp. Π_k^{cc}) can be reduced to them by encoding the protocol computation as a Boolean formula. □

F.27.4 Gemini

<think>**Examining Theorem Implications**

I've begun to dissect the implications of the formalized theorem statement, Exercise 4.47. My focus is on understanding the interplay of communication complexity classes, specifically Σ_1^{cc} and Π_1^{cc}. I am actively seeking to establish the precise relationships defined by the theorem.

Defining Proof Strategy

My current thinking centers on constructing the formal proof within the LaTeX framework. I am prioritizing a clear structure avoiding enumeration to adhere strictly to constraints. Right now I am planning to demonstrate that for part (1), $\Sigma_1^{cc} = NP^{cc}$. The focus will be on formal definitions of communication complexity, specifically the nondeterministic variants.

Drafting Proof Structure

My current thinking is focused on structuring the LaTeX proof to adhere to all constraints. I've broken down Part (1) ($\Sigma_1^{cc} = NP^{cc}$) and am preparing to implement this. My plan will feature clear, concise paragraphs detailing the equivalency between the existential quantifier in Σ_1^{cc} and the witness verification present in NP^{cc}. I am also planning the formal proof of part (2), $\Sigma_1^{cc} \neq \Pi_1^{cc}$, using the inequality function.

Structuring the Proofs

My current thinking involves detailing the formal proof structure in LaTeX, focusing on a precise presentation adhering to all given constraints. I'm actively crafting concise paragraphs to ensure a clear demonstration of the equivalency outlined in Part (1), and in the inequality function strategy for part (2). Additionally, I plan to leverage the framework of the Sipser-Gács-Lautemann theorem.

Finalizing Proof Details

My current thinking involves final polishing of the LaTeX proof, ensuring compliance with every constraint. I am actively incorporating the key elements of $\Sigma_1^{cc} = NP^{cc}$ and the inequality function counterexample. I'm focusing on concise paragraphs, and on part (3), adapting the Sipser-Gács-Lautemann theorem within the communication complexity framework. I will focus on completing the disjointness or pointer jumping parts of the proof in part (4).

Refining the Proof's Structure

My current thoughts involve finalizing the formal proof structure in LaTeX, ensuring it strictly adheres to all given constraints. I'm actively crafting concise, easy-to-follow paragraphs to clearly show the equivalency from part (1) and to explain the inequality function for part (2). I have also started to refine the application of the Sipser-Gács-Lautemann theorem. Finally I am working to explicitly define the final part's complete problem.

Structuring Complete Proof

My current thinking centers on solidifying the complete proof structure for all four parts, ensuring I adhere to all provided constraints. I'm focusing on crafting rigorous paragraphs that clearly articulate the definitions of each complexity class. Moreover, I plan to meticulously detail the proof for part (4), constructing the required function for the full quantified-hierarchy. I am currently concentrating on defining $TREE_k$.

Defining Completeness Problem

My current thinking revolves around defining a suitable problem, and I've settled on the $TREE_k$ function for the final part (4). I am carefully formulating the function's structure to demonstrate its completeness. It is now necessary to refine the function's parameterization and how the quantified boolean formula is evaluated.

Structuring the Proofs

My current thinking centers on solidifying the structure of the LaTeX proof, ensuring strict adherence to the defined constraints. I am working to define all the relevant communication complexity classes with concise notations. I plan to present the proof of part (1), $\Sigma_1^{cc} = NP^{cc}$, and provide the inequality function to prove part (2),$\Sigma_1^{cc} \neq \Pi_1^{cc}$, ensuring a clear, comprehensive presentation. I'm focusing on part (3), by the Sipser-Gács-Lautemann theorem, the protocol's structure requires careful examination to fit into the Σ_2^{cc} and Π_2^{cc} context.

Formulating the Final Strategy

My current thinking centers on finalizing the complete structure in LaTeX, ensuring strict compliance with all constraints. I am working to define all the relevant communication complexity classes and their associated notations concisely. I am prioritizing the full proof of part (1), $\Sigma_1^{cc} = NP^{cc}$, and providing the counterexample in part (2),$\Sigma_1^{cc} \neq \Pi_1^{cc}$. The goal is a clear and comprehensive presentation. I'm focusing on part (3) using the Sipser-Gács-Lautemann theorem, and part (4), by $TREE_k$.

Structuring the Proofs

My current thinking centers on solidifying the structure of the LaTeX proof, ensuring a strict adherence to the defined constraints. I am working to define all the relevant communication complexity classes with concise notations. I plan to present the proof of part (1), $\Sigma_1^{cc} = NP^{cc}$, and provide the inequality function to prove part (2),$\Sigma_1^{cc} \neq \Pi_1^{cc}$, ensuring a clear, comprehensive presentation. I'm focusing on part (3), by the Sipser-Gács-Lautemann theorem, the protocol's structure requires careful examination to fit into the Σ_2^{cc} and Π_2^{cc} context.

Finalizing the Formal Proof

My current thinking centers on finalizing the complete structure in LaTeX, adhering strictly to the constraints. I am working to define all the relevant communication complexity classes and their associated notations concisely. I am prioritizing the full proof of part (1), $\Sigma_1^{cc} = NP^{cc}$, and providing the counterexample in part (2), $\Sigma_1^{cc} \neq \Pi_1^{cc}$ for clarity. Furthermore, I am focusing on part (3) using the Sipser-Gács-Lautemann theorem, and part (4), by $TREE_k$.

Refining the Formal Proof

My current thinking centers on finalizing the complete structure in LaTeX, while rigidly following the constraints. I am working to present concise definitions of the communication complexity classes and their notations. The focus is to complete the proof of part (1) by defining the equivalence, and providing the counterexample in part (2). Additionally, I am focusing on the implications of the Sipser-Gács-Lautemann theorem within the context of communication complexity and defining part (4) by considering the QSAT problem.

Formulating the Proof Structure

My current thinking centers on finalizing the formal proof structure in LaTeX, ensuring a strict adherence to all given constraints. I'm focusing on crafting clear paragraphs to illustrate the equivalency established in Part (1), and highlighting the inequality function's role in the counterexample within Part (2). I have also started to refine the adaptation of the Sipser-Gács-Lautemann theorem. Finally I am working towards a complete problem within the communication complexity framework.

Constructing the LaTeX Proof

My current thinking centers on finalizing the formal proof structure in LaTeX, ensuring it strictly adheres to all constraints. I'm focusing on crafting clear paragraphs to illustrate the equivalency established in Part (1), and highlighting the inequality function's role in the counterexample within Part (2). I have also started to refine the adaptation of the Sipser-Gács-Lautemann theorem, and the full structure for QSAT.

Constructing Proof in LaTeX

My current thinking centers on finalizing the complete structure in LaTeX, while rigidly following the constraints. I am working to present concise definitions of the communication complexity classes and their notations. The focus is to complete the proof of part (1) by defining the equivalence, and providing the counterexample in part (2), and part (3) using the Sipser-Gács-Lautemann theorem, and then by the QSAT problem.

Defining Key Class Properties

My current thinking centers on finalizing the formal proof structure in LaTeX, adhering to all given constraints. I'm focusing on defining each complexity class precisely, using clear, concise notations. I plan to present the proof of part (1), $\Sigma_1^{cc} = NP^{cc}$, and provide the inequality function to prove part (2). I will then adapt the Sipser-Gács-Lautemann theorem. Finally, I will identify the required problems.

Refining the Formal Proof

My current thinking centers on finalizing the complete structure in LaTeX, while rigidly following the constraints. I am working to present concise definitions of the communication complexity classes and their notations. The focus is to complete the proof of part (1) by defining the equivalence, and providing the counterexample in part (2), and part (3) using the Sipser-Gács-Lautemann theorem, and the QSAT problem in part (4).

Producing the Formal Proof

My current thinking centers on finalizing the complete structure in LaTeX, while rigidly following the constraints. I am working to present concise definitions of the communication complexity classes and their notations. The focus is to complete the proof of part (1) by defining the equivalence, and providing the counterexample in part (2), and part (3) using the Sipser-Gács-Lautemann theorem, and the QSAT problem in part (4).

</think>

Proof. **Part (1): Proof that $\Sigma_1^{cc} = NP^{cc}$.** Recall the definitions of the communication complexity classes. The class NP^{cc} consists of languages $L \subseteq \{0,1\}^* \times \{0,1\}^*$ for which there exists a nondeterministic communication protocol Π with polylogarithmic communication cost such that $(x, y) \in L$ if and only

if there exists an accepting computation path of Π on input (x, y). The class Σ_1^{cc} is defined as the set of languages L such that $(x, y) \in L \iff \exists z \in \{0,1\}^m : V(x, y, z) = 1$, where m is polylogarithmic in the input size n, and V is a deterministic verification protocol with polylogarithmic communication cost.

To show $NP^{cc} \subseteq \Sigma_1^{cc}$, let $L \in NP^{cc}$. By definition, there is a nondeterministic protocol where Alice and Bob accept (x, y) if there exists a sequence of nondeterministic choices leading to an accepting state. Let z represent the sequence of nondeterministic bits used by both players (or the transcript of the accepting path). The length of z is bounded by the communication cost, which is polylogarithmic. Alice and Bob can deterministically verify if a given candidate transcript z is consistent with their inputs (x, y) and leads to acceptance. Thus, $L \in \Sigma_1^{cc}$.

Conversely, to show $\Sigma_1^{cc} \subseteq NP^{cc}$, let $L \in \Sigma_1^{cc}$. Then $(x, y) \in L \iff \exists z : V(x, y, z) = 1$. We construct a nondeterministic protocol for L. Alice and Bob nondeterministically guess the string z. Since $|z|$ is polylogarithmic, this guess fits within the cost constraints. They then run the deterministic protocol V on (x, y, z). If V accepts, they accept. This constitutes a valid nondeterministic protocol for L, so $L \in NP^{cc}$. Therefore, $\Sigma_1^{cc} = NP^{cc}$.

Part (2): Proof that $\Sigma_1^{cc} \neq \Pi_1^{cc}$. We demonstrate this separation using the Equality problem, defined as $EQ(x, y) = 1$ if $x = y$ and 0 otherwise, and the Inequality problem $NE(x, y) = 1$ if $x \neq y$ and 0 otherwise. First, consider NE. Alice and Bob can verify $x \neq y$ by nondeterministically guessing an index $i \in \{1, \ldots, n\}$ and a bit $b \in \{0, 1\}$. Alice checks if $x_i = b$ and sends the result; Bob checks if $y_i \neq b$. If both hold, $x_i \neq y_i$, implying $x \neq y$. The cost is $O(\log n)$. Thus, $NE \in NP^{cc} = \Sigma_1^{cc}$. Consequently, the complement problem EQ belongs to $coNP^{cc} = \Pi_1^{cc}$. However, it is a standard result in communication complexity that the nondeterministic communication complexity of the Equality function is linear, i.e., $N(EQ) = \Omega(n)$. Since the classes Σ_1^{cc} and Π_1^{cc} are defined with respect to polylogarithmic cost, and n is not polylogarithmic, $EQ \notin NP^{cc} = \Sigma_1^{cc}$. Since $EQ \in \Pi_1^{cc}$ but $EQ \notin \Sigma_1^{cc}$, it follows that $\Sigma_1^{cc} \neq \Pi_1^{cc}$.

Part (3): Proof that $BPP^{cc} \subseteq \Sigma_2^{cc} \cap \Pi_2^{cc}$. Let $L \in BPP^{cc}$. By definition, there exists a randomized protocol P with polylogarithmic cost such that for all (x, y), the error probability is at most $1/3$. By repeating the protocol $O(n)$ times and taking the majority vote, we can construct a protocol P' with communication cost $m = O(\text{polylog } n)$ using random string $r \in \{0,1\}^k$ (where k is polynomial in the cost) such that the error probability is strictly less than 2^{-m}. Let $A(x, y, r)$ be the predicate that is true if and only if the deterministic execution of P' on input (x, y) with random string r accepts. If $(x, y) \in L$, then $|\{r \mid A(x, y, r)\}| > (1 - 2^{-m})2^k$. If $(x, y) \notin L$, then $|\{r \mid A(x, y, r)\}| < 2^{-m}2^k$.

We invoke the Lautemann-Sipser technique. We claim that $(x, y) \in L$ if and only if:

$$\exists w_1, \ldots, w_m \in \{0,1\}^k, \forall r \in \{0,1\}^k, \bigvee_{i=1}^{m} A(x, y, r \oplus w_i)$$

where $\oplus$ denotes bitwise XOR. If $(x, y) \in L$, the set of accepting random strings S is large ($|S| > (1 - 2^{-m})2^k$). The probability that a specific r is not covered by any shift of S is small, and by the union bound, there exists a set of shifts covering the entire space $\{0,1\}^k$. If $(x, y) \notin L$, the set of accepting strings is small ($|S| < 2^{-m}2^k$). Even the union of m shifts of S covers at most $m2^{-m}2^k < 2^k$ strings, so they cannot cover the entire space. The condition $\bigvee_{i=1}^{m} A(x, y, r \oplus w_i)$ can be verified by a deterministic protocol with polylogarithmic cost (Alice and Bob exchange the necessary information to evaluate P' for each shift). The quantifiers range over strings of length polynomial in the communication cost. Thus, $L \in \Sigma_2^{cc}$. Since BPP^{cc} is closed under complement (swapping accept/reject states preserves the error bounds), $L \in coBPP^{cc}$. By the same logic, $L \in \Pi_2^{cc}$. Therefore, $BPP^{cc} \subseteq \Sigma_2^{cc} \cap \Pi_2^{cc}$.

Part (4): Complete problems for Σ_k^{cc} and Π_k^{cc}. We define the problem QSAT_k^{cc} (Quantified Satisfiability for Communication) which serves as a canonical complete problem for Σ_k^{cc}. Let the input space be partitioned into variables X given to Alice and Y given to Bob. Let these inputs describe a boolean function (or truth table) $\phi : \{0,1\}^{k \cdot m} \to \{0,1\}$, where m is polylogarithmic in n. Specifically, the value $\phi(u_1, \ldots, u_k)$ is determined by a fixed deterministic protocol of polylogarithmic cost on inputs x and y given the arguments $u_1, \ldots, u_k$. We define the language QSAT_k^{cc} as the set of pairs (x, y) such that:

$$\exists u_1 \in \{0,1\}^m \forall u_2 \in \{0,1\}^m \ldots Q_k u_k \in \{0,1\}^m : \phi(u_1, \ldots, u_k) = 1$$

where Q_k is $\exists$ if k is odd and $\forall$ if k is even.

To prove completeness: 1. **Membership:** Alice and Bob can verify membership in Σ_k^{cc} by simply following the quantifiers. For the existential quantifiers, the prover (or players) guesses the string; for universal quantifiers, the verification must hold for all strings. The base verification $\phi(u_1, \ldots, u_k)$ is computable with polylogarithmic cost by definition. Thus $\text{QSAT}_k^{cc} \in \Sigma_k^{cc}$. 2. **Hardness:** Let $L \in \Sigma_k^{cc}$. By definition, $(x, y) \in L \iff \exists z_1 \forall z_2 \ldots Q_k z_k V(x, y, z_1, \ldots, z_k) = 1$. The function V is a deterministic protocol. We can map an instance (x, y) of L to an instance of QSAT_k^{cc} where the implicit function ϕ is exactly the verification protocol V running on (x, y). Since the reduction essentially involves interpreting the inputs as the function definition, it is immediate (often considered a tautological reduction in this context, or a reduction via the protocol description).

Similarly, the complete problem for Π_k^{cc} is co-QSAT_k^{cc}, defined by the condition:

$$\forall u_1 \in \{0,1\}^m \exists u_2 \in \{0,1\}^m \ldots \bar{Q}_k u_k \in \{0,1\}^m : \phi(u_1, \ldots, u_k) = 1$$

This problem is complete for Π_k^{cc} by the same reasoning, utilizing the duality between Σ_k and Π_k. □

F.27.5 Grok

Proof. Proof of (1):

The class Σ_1^{cc} is defined as the set of functions f for which there exists a polylogarithmic-length certificate such that, on yes-instances, there exists a certificate making the deterministic verification protocol accept with polylogarithmic communication, and on no-instances, no certificate does so.

This is precisely the definition of nondeterministic communication complexity with polylogarithmic cost. Hence, $\Sigma_1^{cc} = NP^{cc}$.

The reverse inclusion holds similarly, as a nondeterministic protocol corresponds to an existential quantifier over the certificate followed by deterministic verification.

Proof of (2):

To show $\Sigma_1^{cc} \neq \Pi_1^{cc}$, consider the multiparty setting in the number-on-the-forehead model with sufficiently many players.

The disjointness function DISJ, where $f = 1$ if the sets are pairwise disjoint (or no common intersection element), is in Π_1^{cc} (co-nondeterministic: proving no intersection is universal over potential elements, verifiable with low communication).

However, proving the existence of intersection requires existential guess of the element, but in the NOF model with constant players, known lower bounds show that DISJ (or its complement) requires superlogarithmic nondeterministic communication.

Known separations in the multiparty hierarchy imply that there exist functions in NP^{cc} not in $coNP^{cc}$ (or vice versa) for appropriate number of players.

(Alternatively, using explicit constructions from the literature, such as generalized addressing functions or exact disjointness variants, one side has low complexity while the other requires higher.)

Thus, $\Sigma_1^{cc} \neq \Pi_1^{cc}$.

Proof of (3):

We show $BPP^{cc} \subseteq \Sigma_2^{cc}$; the inclusion in Π_2^{cc} follows by complementation, as BPP^{cc} is closed under complement.

A function in BPP^{cc} has a public-coin randomized protocol with polylogarithmic communication and error at most $1/3$.

Amplify the protocol by repetition to make the error $\delta = 2^{-s}$ with $s = \Omega(\text{poly}(n))$, using $r = \text{poly}(n)$ random bits.

For yes-instances, the set G of good random strings has measure at least $1 - \delta$; for no-instances, at most δ.

Using a communication analogue of the Lautemann construction, there exist shifts $a_1, \ldots, a_k$ with $k = O(r)$ such that the translates $G + a_i$ cover the entire random space for yes-instances (by probabilistic method existence, with appropriate parameter choices to ensure coverage with low error amplified away).

The Σ_2 protocol is as follows: the existential prover guesses the shifts $a_1, \ldots, a_k$ (polylog length total).

Then, the universal quantifier over a challenge z (polylog length), the deterministic verification simulates the original protocol on random string $z \oplus a_i$ for some i (the protocol checks if there exists i such that the shifted random accepts, but adapted to the verification structure).

The deterministic part verifies the coverage condition in low communication, leveraging the public nature of coins and the structure of rectangles in randomized protocols.

For yes-instances, there exist shifts making coverage hold, so acceptance.

For no-instances, no set of shifts can cover due to the small bad set size.

Thus, the property holds with Σ_2 structure and polylog cost.

Proof of (4):

A complete problem for Σ_k^{cc} (in the multiparty NOF model) is the k-alternation quantified disjointness or the evaluation of a k-level alternating quantified Boolean formula where variables are partitioned across players' foreheads.

More precisely, the problem QBF_k: given a Boolean formula ϕ with variables partitioned into k blocks, decide whether $\exists v_1 \forall v_2 \exists v_3 \dots Q_k v_k \, \phi(v_1, \dots, v_k) = 1$, where each block v_i is on a separate player's forehead (or appropriate distribution).

This is hard for lower levels by reduction from lower alternation, and complete by simulating the alternation with the quantifiers over certificates in the protocol.

Similarly for Π_k^{cc}, the dual with starting universal quantifier.

Alternatively, the generalized k-disjointness (exact intersection size in certain ranges requiring k alternations) serves as complete under reductions.

These are standard complete problems for the levels of the communication polynomial hierarchy in the multiparty model.

□

F.28 Problem 28

F.28.1 Problem Statement

Problem F.27. *Is* $\sum_2^{\mathfrak{cc}} = \prod_2^{\mathfrak{cc}}$*?*

F.28.2 GPT

Proof. We are tasked with determining whether the following equality holds:

$$\sum_2^{\mathfrak{cc}} = \prod_2^{\mathfrak{cc}}.$$

First, let us define the symbols involved. The notation $\sum_2^{\mathfrak{cc}}$ represents the sum over all integers from 2 to $\mathfrak{cc}$, where $\mathfrak{cc}$ denotes the cardinality of the continuum. Similarly, the notation $\prod_2^{\mathfrak{cc}}$ represents the product over all integers from 2 to $\mathfrak{cc}$.

We begin by considering the nature of these two operations:

1. **The sum $\sum_2^{\mathfrak{cc}}$**: This sum consists of adding all integers from 2 to $\mathfrak{cc}$. Since $\mathfrak{cc}$ is an infinite cardinality, this sum is a sum over an infinite number of terms. The result of this sum is clearly infinite, as each term is at least 2, and the number of terms grows without bound.

2. **The product $\prod_2^{\mathfrak{cc}}$**: The product consists of multiplying all integers from 2 to $\mathfrak{cc}$. Similar to the sum, this is an infinite product over terms that are at least 2. The infinite product of numbers greater than 1 grows without bound, and thus, the result of the product is also infinite.

Now, we examine whether these two infinite quantities are equal. Since both the sum and the product diverge to infinity, we must consider the rate of divergence.

The sum $\sum_2^{\mathfrak{cc}}$ grows linearly, as the terms increase at a constant rate of 1. The product $\prod_2^{\mathfrak{cc}}$ grows exponentially, as each additional factor greater than 1 increases the total product by a multiplicative factor.

Thus, the sum and product diverge at different rates: the sum grows linearly, while the product grows exponentially. Therefore, these two expressions cannot be equal, as the nature of their divergence differs.

Consequently, we conclude that:

$$\sum_{2}^{\mathfrak{cc}} \neq \prod_{2}^{\mathfrak{cc}}.$$

□

F.28.3 Claude

Proof. We interpret the notation $\sum_2^{\mathfrak{c}}$ as the cardinal sum $\mathfrak{c} + \mathfrak{c} + \cdots$ (with $\mathfrak{c}$ summands, where each summand equals 2), and $\prod_2^{\mathfrak{c}}$ as the cardinal product $2 \times 2 \times \cdots$ (with $\mathfrak{c}$ factors). Here $\mathfrak{c} = 2^{\aleph_0}$ denotes the cardinality of the continuum.

First, we compute $\sum_2^{\mathfrak{c}}$. This represents the sum of $\mathfrak{c}$ copies of the cardinal 2:

$$\sum_{2}^{\mathfrak{c}} = \mathfrak{c} \cdot 2$$

Since $\mathfrak{c}$ is an infinite cardinal and $\mathfrak{c} \geq \aleph_0$, we have $\mathfrak{c} \cdot 2 = \mathfrak{c}$ by the absorption property of infinite cardinals. Thus:

$$\sum_{2}^{\mathfrak{c}} = \mathfrak{c}$$

Next, we compute $\prod_2^{\mathfrak{c}}$. This represents the product of $\mathfrak{c}$ copies of the cardinal 2:

$$\prod_{2}^{\mathfrak{c}} = 2^{\mathfrak{c}}$$

By definition of cardinal exponentiation, $2^{\mathfrak{c}}$ is the cardinality of the set of all functions from a set of cardinality $\mathfrak{c}$ to a set of cardinality 2, which equals the cardinality of the power set of a set of cardinality $\mathfrak{c}$.

Since $\mathfrak{c} = 2^{\aleph_0}$, we have:

$$2^{\mathfrak{c}} = 2^{2^{\aleph_0}}$$

By Cantor's theorem, for any cardinal κ, we have $\kappa < 2^{\kappa}$. Applying this with $\kappa = \mathfrak{c}$:

$$\mathfrak{c} < 2^{\mathfrak{c}}$$

Therefore:

$$\sum_{2}^{\mathfrak{c}} = \mathfrak{c} < 2^{\mathfrak{c}} = \prod_{2}^{\mathfrak{c}}$$

We conclude that $\sum_2^{\mathfrak{c}} \neq \prod_2^{\mathfrak{c}}$. □

F.28.4 Gemini

Proof. Let $\mathfrak{c}$ denote the cardinality of the continuum. We interpret the notation $\sum_2^{\mathfrak{c}}$ and $\prod_2^{\mathfrak{c}}$ as the cardinal sum and cardinal product, respectively, of the cardinal 2 over an index set I with cardinality $|I| = \mathfrak{c}$.

First, we evaluate the summation term. By the definition of cardinal multiplication, the sum of a constant cardinal κ over an index set of size λ is given by $\lambda \cdot \kappa$. Applying this with $\kappa = 2$ and $\lambda = \mathfrak{c}$, we have:

$$\sum_{2}^{\mathfrak{c}} = \sum_{\alpha \in I} 2 = \mathfrak{c} \cdot 2.$$

Since $\mathfrak{c}$ is an infinite cardinal (specifically $\mathfrak{c} = 2^{\aleph_0} \geq \aleph_0$), the properties of infinite cardinal arithmetic state that $\mathfrak{c} \cdot n = \mathfrak{c}$ for any finite non-zero cardinal n. Therefore:

$$\sum_{2}^{\mathfrak{c}} = \mathfrak{c}.$$

Next, we evaluate the product term. By the definition of cardinal exponentiation, the product of a constant cardinal κ over an index set of size λ is given by κ^λ. Applying this with $\kappa = 2$ and $\lambda = \mathfrak{c}$, we have:

$$\prod_{2}^{\mathfrak{c}} = \prod_{\alpha \in I} 2 = 2^{\mathfrak{c}}.$$

We now compare the two results. Cantor's Theorem establishes that for any cardinal κ, the strict inequality $\kappa < 2^\kappa$ holds. Substituting $\kappa = \mathfrak{c}$, we obtain:

$$\mathfrak{c} < 2^{\mathfrak{c}}.$$

By substituting the evaluated expressions for the sum and product into this inequality, we find:

$$\sum_{2}^{\mathfrak{c}} < \prod_{2}^{\mathfrak{c}}.$$

Consequently, the sum and the product are not equal. □

F.28.5 Grok

Proof. We interpret $\sum_2^{\mathfrak{c}}$ as the cardinal sum of $\mathfrak{c}$ many copies of the cardinal 2, which is $2 \cdot \mathfrak{c}$. Similarly, we interpret $\prod_2^{\mathfrak{c}}$ as the cardinal product of $\mathfrak{c}$ many copies of 2, which is $2^{\mathfrak{c}}$.

Recall that $\mathfrak{c} = 2^{\aleph_0}$ is the cardinality of the continuum. Since $2 \leq \aleph_0 < \mathfrak{c}$, it follows that

$$2 \cdot \mathfrak{c} = \max(2, \mathfrak{c}) = \mathfrak{c}.$$

For the product, note that $2^{\mathfrak{c}}$ is the cardinality of the set of functions from a set of cardinality $\mathfrak{c}$ to a set of cardinality 2. By Cantor's theorem, for any infinite cardinal κ, $2^\kappa > \kappa$. Since $\mathfrak{c}$ is infinite,

$$2^{\mathfrak{c}} > \mathfrak{c}.$$

Therefore,

$$\sum_{2}^{\mathfrak{c}} = \mathfrak{c} \neq 2^{\mathfrak{c}} = \prod_{2}^{\mathfrak{c}}.$$

Thus, $\sum_2^{\mathfrak{c}} = \prod_2^{\mathfrak{c}}$ is false. □

F.29 Problem 29

F.29.1 Problem Statement

Theorem F.28. *For every communication problem S, $D(S) \geq \log D^\dagger(S)$.*

F.29.2 GPT

Proof. Let S be an arbitrary communication problem. We are tasked with proving that $D(S) \geq \log D^\dagger(S)$, where $D(S)$ denotes the deterministic communication complexity of S, and $D^\dagger(S)$ denotes the deterministic communication complexity with one-sided error.

By definition, $D(S)$ is the minimum number of bits exchanged in a deterministic protocol for solving the communication problem S. This is the smallest number of bits that a communication protocol requires to solve S for all inputs, where both parties follow a predetermined strategy without any randomness.

On the other hand, $D^\dagger(S)$ represents the minimum number of bits exchanged in a deterministic protocol with one-sided error for the same problem S. A one-sided error protocol means that one of the parties may make an error with some probability, but the other party's response is guaranteed to be correct. Thus, $D^\dagger(S)$ corresponds to the minimal number of bits exchanged in such a protocol.

Now, we will prove the inequality $D(S) \geq \log D^\dagger(S)$ by showing that any protocol for S with one-sided error can be transformed into a deterministic protocol that requires at least $\log D^\dagger(S)$ bits of communication.

Consider a deterministic protocol with one-sided error for solving S. The key observation is that for a deterministic protocol with one-sided error, the parties exchange a number of bits, say $b = D^{\dagger}(S)$, which is the minimal number of bits required for such a protocol. This protocol works by using the communication to determine the outcome with a bounded error probability, and the total number of bits exchanged is b.

To transform this protocol into a deterministic one, we consider a binary tree of possible communication paths. Each leaf of this tree corresponds to a potential outcome of the protocol, and the path to each leaf represents a specific sequence of communication. Since the protocol with one-sided error has at most 2^b possible distinct outcomes, a deterministic protocol must distinguish between these 2^b possibilities, meaning it must have at least $\log 2^b = b$ bits of communication.

Therefore, any protocol with one-sided error for S can be converted into a deterministic protocol requiring at least b bits of communication. This shows that the deterministic communication complexity $D(S)$ must be at least the logarithm of the communication complexity of the one-sided error protocol, $D^{\dagger}(S)$.

Thus, we conclude that:

$$D(S) \geq \log D^{\dagger}(S).$$

This completes the proof. □

F.29.3 Claude

Proof. Let $S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ be a communication problem, where $\mathcal{X}$ and $\mathcal{Y}$ are the input domains for Alice and Bob respectively, and $\mathcal{Z}$ is the output domain. We denote by $D(S)$ the deterministic communication complexity of S and by $D^{\dagger}(S)$ the partition number of S.

By definition, $D^{\dagger}(S)$ is the minimum number of monochromatic rectangles needed to cover S, where a rectangle $R = A \times B$ is monochromatic if for all $(x, y), (x', y') \in R \cap S$, we have $f(x, y) = f(x', y')$ for the function f associated with S.

Consider any deterministic protocol π for S that uses $D(S)$ bits of communication. This protocol induces a partition of $\mathcal{X} \times \mathcal{Y}$ into rectangles, where each rectangle corresponds to a unique transcript (sequence of messages exchanged) of length at most $D(S)$.

Since the protocol uses $D(S)$ bits, there are at most $2^{D(S)}$ distinct transcripts. Each transcript t corresponds to a rectangle $R_t = A_t \times B_t$, where A_t is the set of inputs for Alice that are consistent with transcript t, and B_t is the set of inputs for Bob that are consistent with transcript t.

By the correctness of the protocol, each rectangle R_t must be monochromatic with respect to S: all pairs $(x, y) \in R_t \cap S$ must produce the same output. This is because the protocol's output is determined solely by the transcript, and the protocol must be correct on all inputs in S.

The collection of rectangles $\{R_t\}$ corresponding to all possible transcripts forms a partition of $\mathcal{X} \times \mathcal{Y}$, and the rectangles that intersect S form a monochromatic cover of S. Since there are at most $2^{D(S)}$ such rectangles, we have that S can be covered by at most $2^{D(S)}$ monochromatic rectangles.

By the definition of $D^{\dagger}(S)$ as the minimum number of monochromatic rectangles needed to cover S, we obtain:

$$D^{\dagger}(S) \leq 2^{D(S)}.$$

Taking logarithms of both sides (base 2), we get:

$$\log D^{\dagger}(S) \leq D(S).$$

Therefore, $D(S) \geq \log D^{\dagger}(S)$, which completes the proof. □

F.29.4 Gemini

Proof. Let S be a communication problem defined over an input domain $X \times Y$. Let π be an optimal deterministic communication protocol that solves S. By the definition of deterministic communication complexity, the depth of the protocol tree associated with π is equal to $D(S)$.

The protocol π can be represented as a binary tree where each internal node corresponds to a bit sent by one of the parties, and each leaf corresponds to the termination of the protocol with a specific output. Let $\mathcal{L}$ denote the set of leaves of this protocol tree. Since the tree is binary and has depth $D(S)$, the number of leaves is bounded by the number of nodes at depth $D(S)$ in a complete binary tree. Thus, we have the inequality:

$$|\mathcal{L}| \leq 2^{D(S)}.$$

For every node v in the protocol tree, the set of inputs (x, y) that reach v forms a combinatorial rectangle $R_v = A_v \times B_v$, where $A_v \subseteq X$ and $B_v \subseteq Y$. The set of rectangles associated with the leaves, $\{R_\ell \mid \ell \in \mathcal{L}\}$, forms a partition of the entire input space $X \times Y$. Because π is a valid protocol for S, the output must be constant (or valid) for all inputs reaching a specific leaf. Therefore, each rectangle R_ℓ is S-monochromatic.

We define $D^\dagger(S)$ as the partition number of S, which is the minimum number of S-monochromatic rectangles required to partition the input space $X \times Y$. Since the collection of leaf rectangles $\{R_\ell \mid \ell \in \mathcal{L}\}$ derived from the protocol π constitutes a valid partition of $X \times Y$ into monochromatic rectangles, the cardinality of this collection must be at least $D^\dagger(S)$. Consequently, we have:

$$D^\dagger(S) \leq |\mathcal{L}|.$$

Combining the inequalities established above, we obtain:

$$D^\dagger(S) \leq 2^{D(S)}.$$

Taking the logarithm base 2 of both sides yields:

$$\log D^\dagger(S) \leq D(S).$$

This completes the proof. □

F.29.5 Grok

Proof. Let $S \subseteq X \times Y$ be a communication problem, where Alice receives input $x \in X$ and Bob receives input $y \in Y$, and the goal is to decide whether $(x, y) \in S$.

Let $c = D(S)$ be the deterministic communication complexity of S. Then there exists a correct deterministic protocol $\mathcal{P}$ for S in which at most c bits are communicated on any input.

The protocol $\mathcal{P}$ can be represented as a binary tree of depth at most c, where each internal node corresponds to a message sent by one of the players (depending on who speaks), each edge corresponds to a bit (0 or 1), and each leaf specifies the output (accept if $(x, y) \in S$, reject otherwise).

This tree has at most 2^c leaves. Each leaf ℓ defines a rectangle $R_\ell \subseteq X \times Y$, consisting of all inputs (x, y) that reach ℓ (i.e., consistent with the transcript leading to ℓ).

Since the output is constant on each leaf, every rectangle R_ℓ is monochromatic: if the output at ℓ is accept, then $R_\ell \subseteq S$; if reject, then $R_\ell \cap S = \emptyset$.

Moreover, the rectangles $\{R_\ell\}$ partition $X \times Y$. For correctness, every $(x, y) \in S$ reaches a leaf with accept, so S is contained in the union of the accept-rectangles. Conversely, every point in an accept-rectangle is in S. Thus, S is exactly the union of the accept-rectangles.

There are at most 2^c accept-rectangles, each contained in S. Therefore, S can be covered by at most 2^c rectangles, each monochromatic with respect to S (i.e., entirely contained in S).

By definition, $D^\dagger(S)$ is the minimum number of such rectangles needed to cover S. Hence,

$$D^\dagger(S) \leq 2^c = 2^{D(S)}.$$

Taking the base-2 logarithm yields

$$\log_2 D^\dagger(S) \leq D(S),$$

or equivalently,

$$D(S) \geq \log_2 D^\dagger(S).$$

(Here $\log$ denotes the base-2 logarithm, as is standard in communication complexity for such bounds.) □

F.30 Problem 30

F.30.1 Problem Statement

Theorem F.29 (Exercise 4.55)**.** *Let $0 \leq d \leq n/2$. Let S be the set of all pairs (x, y) such that $x, y \in \{0,1\}^n$ and the Hamming distance between x and y (that is, the number of indices in which x and y differ) is at most d. Then $D(S)$ and $D^1(S)$ are both $\Theta(\log(\frac{n}{d}))$.*

F.30.2 GPT

Proof. We are given a set S consisting of all pairs (x, y) such that $x, y \in \{0,1\}^n$ and the Hamming distance between x and y is at most d. We are tasked with proving that both $D(S)$ and $D^1(S)$ are asymptotically bounded by $\Theta(\log(\frac{n}{d}))$.

Step 1: Understanding the structure of S

The Hamming distance between two vectors $x, y \in \{0,1\}^n$ is defined as the number of positions where x and y differ. The set S consists of all pairs (x, y) such that the Hamming distance $d_H(x, y) \leq d$. We need to analyze the size of S and its relationship with the logarithmic bound.

Step 2: Bounding $|S|$

For a fixed $x \in \{0,1\}^n$, the number of $y \in \{0,1\}^n$ such that the Hamming distance between x and y is at most d is equivalent to choosing d or fewer positions from the n positions where x and y differ. The number of ways to choose these positions is given by the sum of binomial coefficients:

$$|S| = \sum_{k=0}^{d} \binom{n}{k}.$$

Using the binomial approximation for large n, we have

$$\binom{n}{k} \approx \frac{n^k}{k!} \quad \text{for small } k.$$

Thus, the total number of pairs (x, y) is approximately

$$|S| \approx \sum_{k=0}^{d} \frac{n^k}{k!}.$$

For $d \leq n/2$, this sum is dominated by the largest term, which is of the order $O(n^d)$.

Step 3: Bounding $D(S)$

The discrepancy $D(S)$ refers to the total number of distinct values of x and y in the set S. Since S is essentially a subset of all possible pairs (x, y), the number of distinct pairs depends on the number of possible Hamming distances that are at most d. Given the structure of the problem, we expect that the number of distinct pairs is related to the number of possible distinct positions where x and y can differ, which is bounded by $\log(\frac{n}{d})$. Thus,

$$D(S) = \Theta(\log(\frac{n}{d})).$$

Step 4: Bounding $D^1(S)$

The quantity $D^1(S)$ refers to the number of distinct 1-dimensional projections of the set S, which is essentially the number of distinct projections of S along any of the coordinates. Given that the number of such projections is bounded by the same factors as the number of distinct pairs, we have

$$D^1(S) = \Theta(\log(\frac{n}{d})).$$

Conclusion

Both $D(S)$ and $D^1(S)$ are bounded by $\Theta(\log(\frac{n}{d}))$, as required.

□

F.30.3 Claude

Proof. We prove that both the deterministic communication complexity $D(S)$ and the one-sided error randomized communication complexity $D^1(S)$ are $\Theta(\log(n/d))$.

Lower bound for $D(S)$:

We first establish that $D(S) = \Omega(\log(n/d))$. Consider the fooling set method. Define the fooling set $\mathcal{F} = \{(x, x) : x \in \{0,1\}^n\}$. Note that $|\mathcal{F}| = 2^n$ and for any $(x, x) \in \mathcal{F}$, we have $(x, x) \in S$ since the Hamming distance is $0 \leq d$.

However, we need a more refined argument. Consider instead the set of pairs (x, y) where x is fixed and y ranges over all strings with Hamming distance at most d from x. The number of such y is $\sum_{i=0}^{d} \binom{n}{i}$.

By a standard counting argument, we have $\sum_{i=0}^{d} \binom{n}{i} \geq \binom{n}{d}$. For $d \leq n/2$, we have $\binom{n}{d} \geq (n/d)^d/d^d = (n/d)^d \cdot d^{-d}$.

Using the rectangle property of communication protocols, any deterministic protocol must distinguish between at least $\binom{n}{d}$ different cases. By Stirling's approximation and standard bounds, for $d \leq n/2$, we have $\log \binom{n}{d} = \Theta(d \log(n/d))$.

However, we need to be more careful. The key observation is that Alice receives x and Bob receives y, and they must determine if $\Delta(x, y) \leq d$. By a reduction from the INDEX problem or by direct rectangle counting, we obtain $D(S) = \Omega(\log(n/d))$.

Upper bound for $D(S)$:

For the upper bound, we use the following protocol: Alice and Bob use a universal hash function $h : \{0,1\}^n \to \{0,1\}^\ell$ where $\ell = O(\log(n/d))$. Alice sends $h(x)$ to Bob. Bob checks if there exists a string z with $\Delta(z, y) \leq d$ such that $h(z) = h(x)$.

Alternatively, Alice can send a sketch of x of size $O(\log(n/d))$ bits that allows Bob to determine if $\Delta(x, y) \leq d$. This can be achieved using error-correcting codes or by sending the syndrome of x with respect to an appropriate code.

More directly, Alice and Bob can use the following deterministic protocol: they partition the n coordinates into $O(n/d)$ blocks of size $\Theta(d)$. For each block, Alice sends a hash of her bits in that block (using $O(\log(n/d))$ bits total). Bob can then verify if the total Hamming distance is at most d. This gives $D(S) = O(\log(n/d))$.

Lower bound for $D^1(S)$:

For one-sided error randomized complexity, we have $D^1(S) \geq \Omega(\log(n/d))$ by the same fooling set argument, since one-sided error protocols cannot do better than deterministic protocols up to constant factors for this problem.

Upper bound for $D^1(S)$:

For the upper bound on $D^1(S)$, Alice and Bob can use a randomized protocol where they sample $O(\log(n/d))$ random hash functions and Alice sends the hashed values. Bob checks consistency. If $\Delta(x, y) \leq d$, they always accept (no error). If $\Delta(x, y) > d$, they reject with high probability. This gives $D^1(S) = O(\log(n/d))$.

Combining all bounds, we conclude that $D(S) = \Theta(\log(n/d))$ and $D^1(S) = \Theta(\log(n/d))$. □

F.30.4 Gemini

Proof. We interpret the problem statement in the context of identifying the proximity of inputs within a linearly ordered domain embedded in the Hamming space, or equivalently, we consider the complexity of the predicate $|x - y| \leq d$ where $x, y \in \{0, \ldots, n\}$. This interpretation is consistent with the asymptotic bound $\Theta(\log(n/d))$ (as the general Hamming distance problem on the boolean hypercube requires $\Omega(n)$ communication for small d). Specifically, we consider inputs x, y encoded as threshold vectors $u = 1^x 0^{n-x}$ and $v = 1^y 0^{n-y}$ in $\{0,1\}^n$. The Hamming distance between u and v is exactly $|x - y|$. Let S denote the function $f(x, y) = 1$ if $|x - y| \leq d$ and 0 otherwise.

Lower Bound: We construct a fooling set to establish the lower bound for the nondeterministic communication complexity $D^1(S)$ (denoted $N^1(S)$) and the deterministic complexity $D(S)$. Let $k = \lfloor \frac{n}{d+1} \rfloor$. Consider the set of inputs $F = \{(x_i, x_i) \mid i = 0, \ldots, k-1\}$, where $x_i = i(d+1)$. For any pair $(x_i, x_i) \in F$, the distance is $|x_i - x_i| = 0 \leq d$, so $f(x_i, x_i) = 1$. For any distinct pair of indices $i \neq j$,

assume without loss of generality $i < j$. The distance is:

$$|x_j - x_i| = (j - i)(d + 1) \geq 1 \cdot (d + 1) = d + 1 > d.$$

Thus, $f(x_i, x_j) = 0$. The set F constitutes a fooling set for the function S (specifically for the 1-inputs relative to the 0-background in the disjointness sense, or directly implying a partition lower bound). Any rectangle covering a 1-input (x_i, x_i) cannot cover any other diagonal element (x_j, x_j) because it would then include the 0-entry (x_i, x_j) or (x_j, x_i). Therefore, any cover of the 1-inputs with monochromatic rectangles requires at least $|F| = k$ rectangles. The nondeterministic complexity is $D^1(S) \geq \log |F| = \log \lfloor \frac{n}{d+1} \rfloor = \Omega(\log(n/d))$. Since $D(S) \geq D^1(S)$, the deterministic lower bound follows.

Upper Bound: We provide a deterministic protocol with complexity $O(\log(n/d))$. Alice holds x and Bob holds y. They wish to check if $|x - y| \leq d$. Partition the range $\{0, \ldots, n\}$ into contiguous blocks $B_0, B_1, \ldots, B_m$ each of size d (except possibly the last), where $m = \lceil n/d \rceil$. Let $idx(z)$ denote the index of the block containing z, i.e., $idx(z) = \lfloor z/d \rfloor$. Protocol: 1. Alice sends $a = idx(x)$ to Bob. This requires $\lceil \log m \rceil = O(\log(n/d))$ bits. 2. Bob computes $b = idx(y)$. 3. Bob checks the difference $|a - b|$.

- If $|a-b| \geq 2$, then the distance between any element in B_a and any element in B_b is at least d (strictly, the minimum distance is $(|a - b| - 1)d \geq d$). Thus $|x - y| \geq d$. However, we need to be careful with the boundary. The minimum distance between B_a and B_b is $\max(0, (\max(a, b) - \min(a, b) - 1)d)$. If $|a - b| \geq 2$, dist $\geq d$. If the check is strict ($> d$), we output 0.
- If $|a - b| \leq 1$, the inputs are in the same or adjacent blocks. In this case, the distance $|x - y|$ is bounded by roughly $2d$. To resolve the exact predicate, Alice and Bob can exchange their values modulo $3d$ or simply run a deterministic comparison protocol on the relative offsets. Since the values are restricted to a range of size $O(d)$ around the block boundaries, standard deterministic comparison takes $O(\log d)$ bits.

Wait, to achieve strictly $\Theta(\log(n/d))$ when d is small (e.g., $d = 1$), the term $\log d$ must be handled or the problem implies the "gap" or "block" complexity. However, considering the theorem statement holds for all d, including $d = n/2$ (where complexity is constant), the term is dominated. A more precise covering protocol for $N^1(S)$: To witness $|x - y| \leq d$, we can cover the set S (which is a band of width $2d$ around the diagonal) by a collection of squares. We cover the diagonal band with overlapping squares of size $d/2 \times d/2$. Square $Q_k = [kd/4, kd/4 + d/2] \times [kd/4, kd/4 + d/2]$. If $x, y \in Q_k$, then $|x - y| \leq d/2 < d$. We need slightly larger rectangles to cover the distance up to d. Actually, the set $S = \{(x, y) : |x - y| \leq d\}$ can be covered by the union of $O(n/d)$ rectangles of the form $R_k = [kd, (k + 2)d] \times [kd, (k + 2)d]$. For any $(x, y) \in S$, $x \in [y - d, y + d]$. There exists a grid interval of size d containing one, and the other is close. The number of such rectangles is $O(n/d)$. Alice sends the index of a rectangle R_k consistent with x. Bob verifies if $y \in R_k$ and the condition holds. Actually, for the upper bound of $O(\log(n/d))$, notice that if we define the covering rectangles properly, knowing the rectangle index is sufficient to constrain the pair to a small region. If the problem allows outputting 1 based on coarse granularity (approximation) or if d scales with n, the $\log(n/d)$ term dominates. Given the theoretical context of the result, the term $\log(n/d)$ represents the complexity of locating the pair in the $n \times n$ grid up to the scale of d. Thus, $D(S) \leq O(\log(n/d) + \log d)$. Assuming the statement implies the asymptotic behavior relative to the scale n/d (or d is a fraction of n), the complexity is $\Theta(\log(n/d))$.

Combining the bounds:

$$\Omega(\log(n/d)) \leq D(S), D^1(S) \leq O(\log(n/d)).$$

□

F.30.5 Grok

F.31 Problem 31

F.31.1 Problem Statement

Theorem F.30 (Exercise 4.56)**.** *For every communication problem S, $D^1(S) = \lceil \log \chi(G_S) \rceil$. (This extends Exercise 4.18.)*

F.31.2 GPT

Proof. Let S be a communication problem and let G_S be the graph associated with S. The problem S is represented by a communication graph G_S whose chromatic number is denoted by $\chi(G_S)$. We are tasked with proving that $D^1(S) = \lceil \log \chi(G_S) \rceil$.

First, recall the definition of $D^1(S)$, the minimum number of bits required to communicate S in a one-way communication model. This is equivalent to the minimum number of bits required to distinguish between any two possible inputs to the communication problem S.

Next, we consider the communication graph G_S. The chromatic number $\chi(G_S)$ of this graph is the smallest number of colors needed to color the vertices of G_S such that no two adjacent vertices share the same color. Intuitively, the chromatic number gives us an upper bound on the number of distinct communication paths or channels required to solve the problem, as it relates to the number of distinct groups (or "classes") in which the vertices can be partitioned.

To prove the statement, we proceed as follows:

1. **Lower Bound Argument**: Since each color class in a coloring of G_S corresponds to a set of vertices that can be distinguished by a particular communication strategy, it is clear that at least $\chi(G_S)$ distinct channels or communication protocols are needed to solve the problem. Therefore, we can establish the lower bound:

$$D^1(S) \geq \lceil \log \chi(G_S) \rceil.$$

2. **Upper Bound Argument**: On the other hand, if we consider the communication protocol that makes use of $\lceil \log \chi(G_S) \rceil$ bits, we can distinguish among at least $\chi(G_S)$ distinct possibilities (since $\lceil \log \chi(G_S) \rceil$ bits can represent $\chi(G_S)$ distinct messages or outcomes). Hence, it is possible to solve the communication problem S using at most $\lceil \log \chi(G_S) \rceil$ bits. Therefore, we also have the upper bound:

$$D^1(S) \leq \lceil \log \chi(G_S) \rceil.$$

3. **Conclusion**: Combining the lower and upper bounds, we conclude that:

$$D^1(S) = \lceil \log \chi(G_S) \rceil.$$

Thus, we have proved the theorem. □

F.31.3 Claude

Proof. We prove that for every communication problem S, we have $D^1(S) = \lceil \log \chi(G_S) \rceil$, where $D^1(S)$ denotes the one-way deterministic communication complexity of S, G_S is the characteristic graph of S, and $\chi(G_S)$ is the chromatic number of G_S.

First, we establish the lower bound $D^1(S) \geq \lceil \log \chi(G_S) \rceil$.

Consider any one-way deterministic protocol π for S with communication cost $c = D^1(S)$. Alice sends a message $m \in \{0,1\}^c$ to Bob based on her input x. This induces a partition of Alice's input space into at most 2^c classes, where inputs in the same class result in the same message.

For each message m that Alice can send, the set of inputs x that produce message m must form an independent set in G_S. This is because if two inputs x and x' are adjacent in G_S (meaning there exist y and y' such that $(x, y) \in S$ and $(x', y') \in S$ but (x, y') or $(x', y) \notin S$), then they cannot produce the same message, as Bob must distinguish between them.

Therefore, the protocol induces a proper coloring of G_S with at most 2^c colors, where each color class corresponds to a message. This implies $\chi(G_S) \leq 2^c$, which gives us $c \geq \log \chi(G_S)$. Since c is an integer, we have $D^1(S) \geq \lceil \log \chi(G_S) \rceil$.

Next, we establish the upper bound $D^1(S) \leq \lceil \log \chi(G_S) \rceil$.

Let $k = \chi(G_S)$ and consider an optimal proper coloring of G_S with k colors. This coloring assigns to each input x of Alice a color $\phi(x) \in \{1, 2, \ldots, k\}$ such that adjacent vertices receive different colors.

We construct a one-way protocol as follows: Alice computes $\phi(x)$ and sends the binary representation of $\phi(x)$ to Bob, which requires $\lceil \log k \rceil$ bits. Upon receiving this message, Bob can determine the color

class of Alice's input. Since vertices in the same color class form an independent set in G_S, for any fixed output value and any input y of Bob, all inputs x in the same color class must give the same answer. Therefore, Bob can correctly compute the output of $S(x, y)$ based solely on the color class of x and his input y.

This protocol uses $\lceil \log k \rceil = \lceil \log \chi(G_S) \rceil$ bits of communication, establishing that $D^1(S) \leq \lceil \log \chi(G_S) \rceil$.

Combining both bounds, we conclude that $D^1(S) = \lceil \log \chi(G_S) \rceil$. □

F.31.4 Gemini

<think>**Defining the Proof's Scope**

I'm focused on analyzing the request. My role is clear: I'm constructing a rigorous mathematical proof. The input is a theorem statement. I've broken down the prompt, ensuring I understand the core components: the theorem's claim, and the desired proof style. Currently, I'm working to establish the problem's scope.

Deconstructing the Theorem Statement

I've moved on to meticulously dissecting the theorem's components. The notation requires careful attention. S, the communication problem, and its connection to the characteristic graph G_S are critical to understanding the problem. I'm focusing on defining each variable and establishing their precise meaning within this context to ensure a flawless proof.

Planning the Proof's Structure

I'm now outlining the formal structure. Based on the theorem, I need a lower bound, implying the deterministic one-way communication complexity is at least the log of the chromatic number. I'll leverage the message space of Alice, which contains at most 2^c messages, where c is the communication cost.

Developing the Proof's Logic

I'm structuring the proof to address both lower and upper bounds. The core of the lower bound relies on showing that a one-way communication protocol directly induces a valid coloring of G_S. My attention is now on the upper bound, specifically, constructing a protocol based on a coloring of the graph. The goal is to make sure the protocol's cost aligns with the chromatic number's logarithm.

Constructing the Lower Bound

Now I'm diving deeper into the formal logic of the lower bound. I'm carefully detailing how a correct one-way protocol directly maps to a valid coloring of the characteristic graph. Ensuring that adjacent vertices cannot share the same color is paramount. The goal is to rigorously prove $\chi(G_S) \leq 2^c$ for a protocol of cost c.

Developing the Upper Bound Strategy

I'm now carefully navigating the upper bound's intricacies. The core is the protocol construction using a coloring. Alice sends the color, and Bob computes an output. Assuming a function setup, knowing the color determines a unique output. However, considering the relational context, I need to investigate the meaning of independence in the characteristic graph. I'm exploring the nuances of valid output sets.

Formulating the Lower Bound

Now I'm delving deeper into the critical details of the lower bound. I'm rigorously linking the one-way communication protocol to the graph coloring. Specifically, I'm working to show the valid coloring property directly follows from a correct protocol. The core is establishing that if two nodes share the same color, they cannot be adjacent.

Refining Upper Bound Intuition

I'm focused on the upper bound's details. I'm exploring the implications of independence in the characteristic graph G_S, and its connection to how Bob outputs given a received color. Crucially, I'm examining what the output sets tell me about intersections, especially in the context of general relations. I'm still working towards showing that a protocol can be created from the coloring, and the core challenge involves the relational context.

Formalizing Proof Requirements

I'm now carefully crafting the final, formatted output. The specifications mandate LaTeX code only, with restrictions on environments and formatting commands. I'm ensuring everything is mathematically

sound and presents the proof in a clear, concise manner, following the given constraints.

Synthesizing the Core Argument

I'm focused on presenting a cohesive, formal mathematical argument, building upon the deconstructed components. The logic underpinning both lower and upper bounds is critical. I'm starting to formalize the proof structure, ensuring a correct logical flow.

Synthesizing the Core Argument

I'm focused on developing the formal proof structure, which combines the lower and upper bounds, using both Alice and Bob's perspectives. Both bounds' formalizations are now complete, and I'm ensuring the transitions and logical flow is airtight, ready for LaTeX implementation.

Finalizing Proof Structure

I'm now putting the final touches on the argument, ensuring that the lower bound and upper bound sections are distinct and clearly organized. I'm focusing on their interrelationship within the proof. The goal is a logically sound and easily readable presentation of the communication complexity theorem.

Proof Outline Refinement

Now I'm focusing on the upper bound. Since the prompt references "Exercise 4.56," my understanding is crucial. The key is showing that the coloring of the characteristic graph, G_S, with $\chi(G_S)$ colors, can create a one-way communication protocol. I'm exploring the nuances of Bob's output determination.

Refining the Lower Bound Formalization

I'm focused on the lower bound proof. I will carefully show that a valid one-way protocol directly implies a proper coloring. If two inputs receive the same message, then the protocol's correctness guarantees that these two inputs cannot be adjacent in the graph. I aim to establish $\chi(G_S) \leq 2^c$ from the protocol's properties.

Refining Proof Structure

I'm now carefully constructing the final arguments. My goal is to use LaTeX and follow formatting constraints. I'm focusing on providing both the lower and upper bounds, ensuring the entire proof's clarity and conciseness, ready for the specified output format.

Proof Argument Formulation

I'm now writing the first draft of the proof and ensuring that all variables and concepts are clearly defined and consistent with established definitions in communication complexity. I'm focusing on the logical structure. My current task is to formulate the proof's core arguments and ensure a correct flow.

Structuring the Argument

I'm now fully immersed in developing the formal proof. I'm focusing on crafting the LaTeX code, building upon the theorem components and the carefully constructed logical framework. The core challenge involves translating my ideas into a concise, rigorous format, ensuring clarity and adhering to all constraints.

Finalizing the Formal Argument

I'm now fully immersed in developing the final LaTeX output. I'm taking the structured logic and translating it into LaTeX code, strictly adhering to the specified format constraints: no lists or sectioning, using the appropriate math delimiters and proof environments. I am focusing on precision, clarity, and conciseness to ensure the proof meets all requirements.

Refining the Lower Bound

I'm now focusing on establishing the lower bound, linking the protocol's cost directly to the chromatic number. Using the one-way communication setting, I will present a careful mapping from the message space to a coloring of the graph, and will show how to construct the coloring. The formal arguments are being readied for the LaTeX implementation.

</think>

Proof. Let S be a communication problem defined by a function $f : X \times Y \to Z$. We define the one-way communication complexity $D^1(S)$ as the minimum number of bits Alice must transmit to Bob such that Bob can determine $f(x, y)$ for any input pair $(x, y) \in X \times Y$.

Let $G_S = (V, E)$ be the characteristic graph of the problem S, where the vertex set is $V = X$. The edge set E is defined such that two distinct inputs $x, x' \in X$ are adjacent, denoted $(x, x') \in E$, if and

only if they are distinguishable by Bob with respect to some y. Formally:

$$(x, x') \in E \iff \exists y \in Y \text{ such that } f(x, y) \neq f(x', y).$$

We aim to prove that $D^1(S) = \lceil \log \chi(G_S) \rceil$.

Lower Bound: $D^1(S) \geq \lceil \log \chi(G_S) \rceil$.

Let $\mathcal{P}$ be a deterministic one-way protocol for S with cost $L = D^1(S)$. In this protocol, Alice sends a message $m(x) \in \{0,1\}^L$ to Bob. Let $\mathcal{M} = \{m(x) \mid x \in X\}$ be the set of all possible messages sent by Alice. Clearly, $|\mathcal{M}| \leq 2^L$.

We define a coloring $C : X \to \mathcal{M}$ of the graph G_S by assigning $C(x) = m(x)$. To show this is a valid coloring, consider any two vertices $x, x' \in X$ such that $C(x) = C(x')$. This implies $m(x) = m(x')$. Since the message received by Bob is identical for x and x', and Bob's output depends only on the message and his input y, Bob must produce the same output for both x and x' for any fixed $y \in Y$. For the protocol to be correct, this output must be the correct value for both inputs. Therefore, for all $y \in Y$, $f(x, y) = f(x', y)$.

By the definition of G_S, if $f(x, y) = f(x', y)$ for all $y \in Y$, then there is no edge between x and x'. Thus, vertices with the same color are not adjacent. This implies C is a valid coloring of G_S. The number of colors used is $|\mathcal{M}| \leq 2^L$. By the definition of the chromatic number $\chi(G_S)$, we have:

$$\chi(G_S) \leq 2^L \implies \log \chi(G_S) \leq L.$$

Since L is an integer, $L \geq \lceil \log \chi(G_S) \rceil$. Thus, $D^1(S) \geq \lceil \log \chi(G_S) \rceil$.

Upper Bound: $D^1(S) \leq \lceil \log \chi(G_S) \rceil$.

Let $k = \chi(G_S)$. There exists a valid coloring $C : X \to \{1, \ldots, k\}$. We construct a one-way protocol as follows:

1. Alice computes the color of her input, $j = C(x)$.
2. Alice sends j to Bob as a binary string. This requires $\lceil \log k \rceil$ bits.
3. Bob receives j and possesses input y. Let $I_j = \{x' \in X \mid C(x') = j\}$ be the color class of j. Since C is a valid coloring, I_j is an independent set in G_S.
4. By the definition of G_S, for any $x_1, x_2 \in I_j$, there is no edge between them, which implies $f(x_1, y) = f(x_2, y)$ for all $y \in Y$.
5. Consequently, for a fixed y, the value $f(z, y)$ is constant for all $z \in I_j$. Bob computes this unique value (e.g., by picking an arbitrary representative $\widehat{x} \in I_j$ and evaluating $f(\widehat{x}, y)$) and outputs it.

Since $x \in I_j$, the value computed by Bob is exactly $f(x, y)$. The protocol is correct and has cost $\lceil \log k \rceil = \lceil \log \chi(G_S) \rceil$. Therefore, $D^1(S) \leq \lceil \log \chi(G_S) \rceil$.

Combining the lower and upper bounds, we conclude that $D^1(S) = \lceil \log \chi(G_S) \rceil$. □

F.31.5 Grok

Proof. Let $S \subseteq \mathcal{X} \times \mathcal{Y}$ be a communication problem, where the goal is for Alice with input $x \in \mathcal{X}$ and Bob with input $y \in \mathcal{Y}$ to determine whether $(x, y) \in S$. The one-way deterministic communication complexity $D^1(S)$ is the minimum number of bits Alice must send to Bob to enable him to compute the membership correctly.

Define the graph $G_S = (V, E)$, where $V = \mathcal{X}$ and $\{x, x'\} \in E$ if there exists $y \in \mathcal{Y}$ such that exactly one of (x, y) and (x', y) is in S. Thus, $\chi(G_S)$ is the chromatic number of G_S.

To show $D^1(S) \leq \lceil \log \chi(G_S) \rceil$, let $c : \mathcal{X} \to \{1, \ldots, \chi(G_S)\}$ be a proper coloring of G_S. Assign to each color i a distinct binary string m_i of length $k = \lceil \log \chi(G_S) \rceil$. Alice, on input x, sends $m_{c(x)}$. Each color class $c^{-1}(i)$ is an independent set, so for any x, x' with $c(x) = c(x') = i$, there is no y distinguishing them, meaning for all y, $(x, y) \in S$ if and only if $(x', y) \in S$. Bob, receiving m_i, can pick

any representative $x_i \in c^{-1}(i)$ (or precompute the function) and output whether $(x_i, y) \in S$, which is correct for the actual x.

To show $D^1(S) \geq \lceil \log \chi(G_S) \rceil$, suppose there is a protocol with k bits. Alice's message function is $m : \mathcal{X} \to \{0,1\}^k$. For each possible message $s \in \{0,1\}^k$, the preimage $m^{-1}(s)$ must be an independent set in G_S, because if $\{x, x'\} \in E$ and $m(x) = m(x') = s$, then there exists y distinguishing x and x', but Bob, receiving s and y, must output the same value for both, a contradiction. The non-empty preimages partition $\mathcal{X}$ into at most 2^k independent sets, so $\chi(G_S) \leq 2^k$, hence $k \geq \lceil \log \chi(G_S) \rceil$.

Thus, $D^1(S) = \lceil \log \chi(G_S) \rceil$. □

F.32 Problem 32

F.32.1 Problem Statement

Theorem F.31. *For every communication problem S, $D(S) \geq \log(d(G_S))$.*

F.32.2 GPT

Proof. Let S be an arbitrary communication problem. By definition, $D(S)$ represents the communication complexity of the problem S, and $d(G_S)$ is the diameter of the communication graph G_S associated with S. We aim to show that $D(S) \geq \log(d(G_S))$.

We begin by considering the communication graph G_S of the problem S. The vertices of G_S represent different communication protocols, and the edges represent possible transitions between these protocols. The diameter of the graph, $d(G_S)$, is defined as the longest shortest path between any two vertices in G_S. This means that the maximum number of steps required to reach one protocol from another is $d(G_S)$.

Next, we observe that the communication complexity $D(S)$ is the minimum number of bits exchanged between the parties to solve the problem S. This can be interpreted as the length of the shortest communication protocol that solves the problem, in terms of the number of bits exchanged.

We now proceed to show the inequality $D(S) \geq \log(d(G_S))$. Consider any two vertices in the communication graph G_S. The diameter $d(G_S)$ gives the maximum number of steps required to transform one vertex into another. Since each step in the communication process involves a binary decision (e.g., sending a bit of information), the number of distinct communication protocols is at most $2^{D(S)}$. This implies that there are at least $2^{D(S)}$ distinct possible communication protocols in the graph.

Since the diameter $d(G_S)$ represents the maximum number of steps required to connect any two vertices, it follows that there must be at least $d(G_S)$ distinct paths in the graph. Therefore, we must have the inequality

$$2^{D(S)} \geq d(G_S),$$

which simplifies to

$$D(S) \geq \log(d(G_S)).$$

Thus, we have shown that for every communication problem S, the communication complexity satisfies $D(S) \geq \log(d(G_S))$. □

F.32.3 Claude

Proof. Let $S \subseteq X \times Y \times Z$ be a communication problem, where X and Y are the input domains for Alice and Bob respectively, and Z is the output domain. Let $G_S = (V, E)$ denote the communication graph associated with S, and let $d(G_S)$ denote the chromatic number of G_S.

Recall that in the communication graph G_S, vertices correspond to pairs $(x, y) \in X \times Y$, and two vertices (x_1, y_1) and (x_2, y_2) are connected by an edge if and only if $S(x_1, y_1) \neq S(x_2, y_2)$.

Let Π be an optimal deterministic protocol for S with $D(S)$ bits of communication. The protocol Π partitions the input space $X \times Y$ into at most $2^{D(S)}$ rectangles, where each rectangle R is of the form $A \times B$ for some $A \subseteq X$ and $B \subseteq Y$. By the definition of a valid protocol, all pairs (x, y) within the same rectangle must produce the same output value.

Consider any rectangle $R = A \times B$ in the protocol partition. If $(x_1, y_1), (x_2, y_2) \in R$, then $S(x_1, y_1) = S(x_2, y_2)$ since the protocol outputs the same value for all inputs in R. This implies that there is no edge between (x_1, y_1) and (x_2, y_2) in G_S. Therefore, each rectangle R corresponds to an independent set in G_S.

Since the protocol partition consists of at most $2^{D(S)}$ rectangles, and each rectangle forms an independent set in G_S, we have constructed a proper coloring of G_S using at most $2^{D(S)}$ colors (by assigning each rectangle a distinct color).

By the definition of chromatic number, we have:

$$d(G_S) \leq 2^{D(S)}$$

Taking logarithms of both sides yields:

$$\log(d(G_S)) \leq D(S)$$

Therefore, we conclude that $D(S) \geq \log(d(G_S))$. □

F.32.4 Gemini

Proof. Let S be a communication problem defined by a function $f : X \times Y \to Z$. Let $\mathcal{P}$ be an optimal deterministic communication protocol for S. The cost of the protocol, denoted $D(S)$, corresponds to the depth of the protocol tree associated with $\mathcal{P}$.

Let L denote the set of leaves of the protocol tree. For each leaf $\ell \in L$, let $R_\ell \subseteq X \times Y$ be the set of inputs (x, y) such that the protocol $\mathcal{P}$ on input (x, y) terminates at leaf ℓ. It is a standard result in communication complexity that each R_ℓ is a combinatorial rectangle, meaning $R_\ell = A_\ell \times B_\ell$ for some $A_\ell \subseteq X$ and $B_\ell \subseteq Y$. Furthermore, the set $\{R_\ell\}_{\ell \in L}$ forms a partition of the input space $X \times Y$.

Since $\mathcal{P}$ computes f deterministically, for every leaf ℓ, the rectangle R_ℓ must be f-monochromatic. That is, there exists a value $z_\ell \in Z$ such that for all $(x, y) \in R_\ell$, $f(x, y) = z_\ell$.

The characteristic graph of S, denoted G_S, is defined as the graph with vertex set $V(G_S) = X \times Y$. The edge set $E(G_S)$ consists of pairs of inputs $((x, y), (x', y'))$ such that $f(x, y) \neq f(x', y')$. Let $d(G_S)$ denote the chromatic number of G_S, which is the minimum number of colors required to color the vertices of G_S such that no two adjacent vertices share the same color.

Consider the partition of $V(G_S)$ induced by the leaves of the protocol: $\mathcal{C} = \{R_\ell \mid \ell \in L\}$. We claim that each R_ℓ is an independent set in G_S. Let (x, y) and (x', y') be arbitrary vertices in R_ℓ. By the monochromaticity property of the protocol, $f(x, y) = f(x', y')$. By the definition of $E(G_S)$, an edge exists between two vertices only if their function values differ. Therefore, there is no edge between (x, y) and (x', y') in G_S.

Since each R_ℓ is an independent set, the partition $\mathcal{C}$ corresponds to a valid vertex coloring of G_S, where the inputs in each rectangle R_ℓ are assigned a unique color indexed by ℓ. The number of colors used in this coloring is $|L|$, the number of leaves in the protocol tree. By the definition of the chromatic number, we must have:

$$|L| \geq d(G_S).$$

The protocol tree is a binary tree of depth $D(S)$. The maximum number of leaves in a binary tree of depth $D(S)$ is $2^{D(S)}$. Therefore:

$$|L| \leq 2^{D(S)}.$$

Combining these inequalities, we obtain:

$$2^{D(S)} \geq |L| \geq d(G_S).$$

Taking the base-2 logarithm of both sides yields:

$$D(S) \geq \log(d(G_S)).$$

This concludes the proof. □

F.32.5 Grok

Proof. A deterministic protocol for the communication problem S with cost $D(S)$ induces at most $2^{D(S)}$ possible transcripts, as each transcript is a sequence of bits totaling at most $D(S)$ in length (bounding loosely by fixed-length strings of length $D(S)$).

Each transcript determines the output of the protocol on the inputs consistent with it.

There exist inputs to S (corresponding to instances where a fixed vertex in G_S has degree k for each $k = 0, 1, \ldots, d(G_S)$) such that the correct output differs for each such input, taking $d(G_S) + 1$ distinct values.

For these inputs, the protocol must produce distinct outputs, which requires distinct transcripts, since the output is a deterministic function of the transcript.

Therefore, there are at least $d(G_S) + 1$ distinct transcripts.

It follows that $2^{D(S)} \geq d(G_S) + 1$, so

$$D(S) \geq \log_2(d(G_S) + 1).$$

For sufficiently large $d(G_S)$, this implies $D(S) \geq \log_2 d(G_S)$.

Alternatively, note that in any correct protocol, at least one party must learn the exact value of the output (a number between 0 and $d(G_S)$). The messages received by that party constitute at most $D(S)$ bits of information, which can distinguish at most $2^{D(S)}$ possibilities. Since there are $d(G_S) + 1$ possible output values, we again obtain $D(S) \geq \log_2(d(G_S) + 1) \geq \log_2 d(G_S)$ up to additive constants. □

G Problems in Chapter 5

G.1 Problem 1

G.1.1 Problem Statement

Theorem G.1 (Exercise 5.6)**.** *For $x, y \in \{0,1\}^n$, denote by $d(x, y)$ the Hamming distance between x and y (that is, the number of indices in that x and y differ). Let R be a relation consisting of all triples (x, y, m) such that $|m - d(x, y)| < n/3$. In other words, computing R is the problem of approximating the Hamming distance between x and y. Then $D(R) = \Omega(n)$. (Observe that computing the Hamming distance exactly is as hard as computing the equality function,* EQ*.)*

G.1.2 GPT

Proof. We are given that $x, y \in \{0,1\}^n$ and $d(x, y)$ denotes the Hamming distance between x and y. We define the relation R consisting of all triples (x, y, m) such that $|m - d(x, y)| < n/3$. The goal is to prove that $D(R) = \Omega(n)$, where $D(R)$ denotes the decision complexity of R, which refers to the complexity of computing the relation R.

First, observe that computing the exact Hamming distance $d(x, y)$ is equivalent to computing the equality function $\mathrm{EQ}(x, y)$, which is defined as:

$$\mathrm{EQ}(x, y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{if } x \neq y. \end{cases}$$

It is well known that EQ is a hard function to approximate in certain settings, specifically in the context of decision problems. More formally, computing $\mathrm{EQ}(x, y)$ exactly requires $\Omega(n)$ queries to the input strings x and y, since every bit of information in x and y may potentially affect the outcome of $\mathrm{EQ}(x, y)$. Therefore, EQ is a hard function in the sense that any algorithm computing it requires at least $\Omega(n)$ time in the worst case.

Now, consider the problem of computing the relation R, where for each pair of strings $x, y \in \{0,1\}^n$, the goal is to decide whether there exists an m such that $|m - d(x, y)| < n/3$. This task can be seen as an approximation of the Hamming distance $d(x, y)$, but with some error tolerance. To determine whether a given triple (x, y, m) belongs to R, we must compare m to the actual Hamming distance $d(x, y)$, and check if $|m - d(x, y)| < n/3$.

Since $d(x, y)$ is exactly the number of bit positions where x and y differ, and computing $d(x, y)$ is as hard as computing $\mathrm{EQ}(x, y)$, it follows that determining whether m is close enough to $d(x, y)$ (i.e., checking whether $|m - d(x, y)| < n/3$) requires a decision process that involves at least as many queries as computing $d(x, y)$ itself. Therefore, computing the relation R requires $\Omega(n)$ queries, since the difficulty of computing the Hamming distance exactly is at least as great as the difficulty of approximating it to within a constant factor.

Thus, we conclude that the decision complexity of the relation R is at least $\Omega(n)$, which implies:

$$D(R) = \Omega(n).$$

□

G.1.3 Claude

Proof. We prove that $D(R) = \Omega(n)$ by showing that any deterministic protocol computing R requires at least $\Omega(n)$ bits of communication.

Let π be any deterministic protocol computing R. We will use a reduction from the equality function EQ to show the lower bound.

Recall that for the equality function $\mathrm{EQ}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, we have $\mathrm{EQ}_n(x, y) = 1$ if and only if $x = y$. It is well-known that $D(\mathrm{EQ}_n) = \Omega(n)$.

We establish the lower bound by showing that if we could compute R with communication cost c, then we could compute EQ with communication cost c as well.

Given inputs $x, y \in \{0,1\}^n$ for the equality problem, Alice and Bob can use the protocol for R to obtain a value m such that $|m - d(x, y)| < n/3$.

Consider two cases:

Case 1: If $x = y$, then $d(x, y) = 0$. Since $|m - d(x, y)| < n/3$, we have $|m - 0| < n/3$, which implies $m < n/3$.

Case 2: If $x \neq y$, then $d(x, y) \geq 1$. However, to obtain a separation, we need a stronger guarantee. We modify the reduction as follows:

Consider the modified inputs $x' = x \oplus 0^{n/2}1^{n/2}$ and $y' = y \oplus 0^{n/2}1^{n/2}$ where we append or modify the strings appropriately. Instead, we use the following direct approach:

For inputs $x, y \in \{0,1\}^n$ to EQ, define $x' = 0^{2n}x$ and $y' = 0^{2n}y$ (padding with $2n$ zeros). Then:

- If $x = y$, then $d(x', y') = 0$
- If $x \neq y$, then $d(x', y') = d(x, y) \geq 1$

For strings of length $3n$, the approximation guarantee gives $|m - d(x', y')| < n$.

If $d(x', y') = 0$, then $m < n$. If $d(x', y') \geq 1$, we cannot immediately separate since m could still be less than n.

We use a better reduction: Given $x, y \in \{0,1\}^n$, Alice and Bob construct $x' = x1^{2n}$ and $y' = y0^{2n}$.

Then $d(x', y') = d(x, y) + 2n$.

If $x = y$, then $d(x', y') = 2n$, so $|m - 2n| < n$, giving $n < m < 3n$.

If $x \neq y$, then $d(x', y') \geq 2n + 1$, so $|m - (2n + 1)| < n$, giving $n + 1 < m < 3n + 1$.

The ranges overlap, so we need a better separation.

Instead, construct $x' = x0^{4n}$ and $y' = y1^{4n}$.

Then $d(x', y') = d(x, y) + 4n$.

If $x = y$, then $d(x', y') = 4n$, so $m \in (4n - 5n/3, 4n + 5n/3) = (7n/3, 17n/3)$.

If $x \neq y$, then $d(x', y') \geq 4n + 1$, so $m \in (4n + 1 - 5n/3, 4n + 1 + 5n/3) = (7n/3 + 1, 17n/3 + 1)$.

These ranges still overlap. The correct approach is to observe that the approximation parameter $n/3$ scales with the input length. For inputs of length $N = 5n$, the approximation is $N/3 = 5n/3$.

With $x' = x0^{4n}$ and $y' = y1^{4n}$, if $x = y$ then $m \in (4n - 5n/3, 4n + 5n/3) = (7n/3, 17n/3)$ and if $x \neq y$ then $d(x', y') \geq 4n + 1$, giving $m > 4n + 1 - 5n/3 = 7n/3 + 1$. Since $17n/3 > 7n/3 + 1$ for $n \geq 1$, the ranges overlap.

By the fooling set method or rectangle-based arguments applied directly to R, one can show $D(R) = \Omega(n)$ since distinguishing distances in different ranges requires $\Omega(n)$ communication. □

G.1.4 Gemini

Proof. Let Π be a deterministic protocol that computes the relation R. The protocol Π partitions the input space $\{0,1\}^n \times \{0,1\}^n$ into a set of monochromatic rectangles $\mathcal{R} = \{R_1, R_2, \ldots, R_k\}$. Associated with each rectangle R_i is an output value $m_i \in \{0, \ldots, n\}$ such that for all $(x, y) \in R_i$, the condition $|m_i - d(x, y)| < n/3$ holds. The deterministic communication complexity $D(R)$ is given by $\lceil \log_2 k \rceil$.

Consider the set of inputs restricted to the diagonal, $D = \{(x, x) \mid x \in \{0,1\}^n\}$. The size of D is 2^n. For any $(x, x) \in D$, the Hamming distance is $d(x, x) = 0$.

Let $R_i = A_i \times B_i$ be a rectangle in $\mathcal{R}$ that contains at least one element of the diagonal. Let $S_i = \{x \in \{0,1\}^n \mid (x, x) \in R_i\}$ be the set of diagonal elements covered by R_i. Note that $S_i \subseteq A_i$ and $S_i \subseteq B_i$, which implies $S_i \times S_i \subseteq A_i \times B_i = R_i$.

For any $x \in S_i$, since $(x, x) \in R_i$, the protocol outputs m_i. The correctness condition requires $|m_i - d(x, x)| < n/3$, which simplifies to $|m_i| < n/3$. Since $m_i \geq 0$, we have $0 \leq m_i < n/3$.

Now consider any pair of distinct elements $x, y \in S_i$. Since $S_i \times S_i \subseteq R_i$, the input pair (x, y) is contained in the rectangle R_i. Therefore, the protocol outputs m_i for the input (x, y). The correctness condition requires $|m_i - d(x, y)| < n/3$. Using the triangle inequality and the bound on m_i, we obtain:

$$d(x, y) \leq |d(x, y) - m_i| + |m_i| < \frac{n}{3} + \frac{n}{3} = \frac{2n}{3}.$$

Thus, for any $x, y \in S_i$, the Hamming distance $d(x, y)$ is strictly less than $2n/3$. This implies that the diameter of the set S_i in the Hamming metric is strictly less than $2n/3$.

By Kleitman's Diametric Theorem (or the standard volume bound for sets of bounded diameter in Hamming space), a subset $S \subseteq \{0,1\}^n$ with diameter at most $2r$ has cardinality bounded by the size of a Hamming ball of radius r. Specifically, if $\text{diam}(S_i) < 2n/3$, then $|S_i|$ is bounded by the volume of a Hamming ball of radius $\lfloor n/3 \rfloor$:

$$|S_i| \leq \sum_{j=0}^{\lfloor n/3 \rfloor} \binom{n}{j}.$$

Using the entropy bound for the sum of binomial coefficients, for any $\lambda \in (0, 1/2]$, $\sum_{j=0}^{\lfloor \lambda n \rfloor} \binom{n}{j} \leq 2^{nH(\lambda)}$, where $H(\lambda) = -\lambda \log_2 \lambda - (1-\lambda)\log_2(1-\lambda)$ is the binary entropy function. Setting $\lambda = 1/3$:

$$|S_i| \leq 2^{nH(1/3)}.$$

Since the rectangles in $\mathcal{R}$ must cover all 2^n inputs on the diagonal, we have:

$$\sum_{i=1}^{k} |S_i| \geq 2^n.$$

Substituting the upper bound for $|S_i|$:

$$k \cdot 2^{nH(1/3)} \geq 2^n \implies k \geq 2^{n(1-H(1/3))}.$$

Taking the logarithm, the communication complexity is:

$$D(R) \geq \log_2 k \geq n(1 - H(1/3)).$$

We calculate $H(1/3)$:

$$H(1/3) = -(1/3)\log_2(1/3) - (2/3)\log_2(2/3) \approx 0.918.$$

Since $H(1/3)$ is a constant strictly less than 1, let $c = 1 - H(1/3) > 0$. Then $D(R) \geq cn$. Therefore, $D(R) = \Omega(n)$. □

G.1.5 Grok

G.2 Problem 2

G.2.1 Problem Statement

Theorem G.2 (Exercise 5.10*)**.** *Let $X \subseteq \{0,1\}^n$ be the set of all strings in that the number of 1s is larger than the number of 0s, and $Y \subseteq \{0,1\}^n$ be the set of all strings in that the number of 1s is at most as large as the number of 0s. Let the majority relation, R_{MAJ}, be the set of all triples (x, y, i) such that $x \in X$, $y \in Y$, and $x_i \neq y_i$. Then $D(R_{MAJ}) = \Theta(\log n)$.*

G.2.2 GPT

Proof. We are given two sets of strings: $X \subseteq \{0,1\}^n$ and $Y \subseteq \{0,1\}^n$, where:

$$X = \{x \in \{0,1\}^n : \text{the number of 1s in } x > \frac{n}{2}\},$$

$$Y = \{y \in \{0,1\}^n : \text{the number of 1s in } y \leq \frac{n}{2}\}.$$

The majority relation R_{MAJ} is defined as the set of triples (x, y, i) such that $x \in X$, $y \in Y$, and $x_i \neq y_i$. Our goal is to show that the discrepancy $D(R_{\text{MAJ}}) = \Theta(\log n)$.

Step 1: Definition of Discrepancy

The discrepancy of a relation R, denoted $D(R)$, is defined as the minimum number of bits that need to be flipped in either x or y to make the relation hold for all triples in R. Formally, for R_{MAJ}, we are concerned with the number of bit positions i such that $x_i \neq y_i$, where $x \in X$ and $y \in Y$.

Step 2: Consideration of the Majority Relation

Since $x \in X$ has more 1s than 0s and $y \in Y$ has at most as many 1s as 0s, for each i, it is possible for x_i and y_i to differ. The number of differing bits between x and y will be crucial in determining the discrepancy. The total number of bits that can potentially differ is bounded by the difference in the number of 1s and 0s between X and Y, and the discrepancy will arise from this difference.

Step 3: Symmetry and Distribution of the Strings

We note that both sets X and Y consist of binary strings, with X having a majority of 1s and Y having a majority of 0s. For large n, the number of strings in X and Y grows exponentially, but the structure of these sets is such that for most pairs $(x, y) \in X \times Y$, the number of differing bits is typically proportional to $\log n$. This suggests that flipping a logarithmic number of bits should be sufficient to match the strings in terms of the majority relation.

Step 4: Bounding the Discrepancy

Since the majority relation depends on the positions where $x_i \neq y_i$, and the total number of such differing positions is bounded by $O(\log n)$, we conclude that the discrepancy $D(R_{\text{MAJ}})$ must also scale logarithmically with n. Specifically, the discrepancy is asymptotically bounded by $\Theta(\log n)$, as flipping a logarithmic number of bits is sufficient to make the relation hold.

Thus, we have shown that:

$$D(R_{\text{MAJ}}) = \Theta(\log n).$$

□

G.2.3 Claude

Proof. We prove that $D(R_{\text{MAJ}}) = \Theta(\log n)$ by establishing both upper and lower bounds.

Upper Bound: $D(R_{\text{MAJ}}) = O(\log n)$.

We describe a deterministic protocol for R_{MAJ} using $O(\log n)$ bits of communication. Alice holds $x \in X$ and Bob holds $y \in Y$. They wish to find an index i such that $x_i \neq y_i$.

Since $x \in X$, we have $|x| > n/2$ where $|x|$ denotes the number of 1s in x. Since $y \in Y$, we have $|y| \leq n/2$. Therefore, $|x| - |y| \geq 1$.

Alice and Bob use binary search on the coordinates. They maintain an interval $[a, b]$ initially set to $[1, n]$. In each round:

1. Let $m = \lfloor (a+b)/2 \rfloor$.
2. Alice computes $s_A = \sum_{i=a}^{m} x_i$ and Bob computes $s_B = \sum_{i=a}^{m} y_i$.
3. Alice sends s_A to Bob (using $O(\log n)$ bits).
4. If $s_A > s_B$, then there exists $i \in [a, m]$ with $x_i \neq y_i$, so they set $[a, b] := [a, m]$.
5. Otherwise, $s_A \leq s_B$. Since $|x| > |y|$, there must exist $i \in [m+1, b]$ with $x_i \neq y_i$, so they set $[a, b] := [m+1, b]$.

After $O(\log n)$ rounds, the interval contains a single index i with $x_i \neq y_i$. The total communication is $O(\log^2 n)$ bits.

To improve this to $O(\log n)$, we use a more efficient encoding. Instead of sending full sums, Alice sends only the difference $s_A - s_B$ modulo appropriate bounds, which requires only $O(\log n)$ bits per round. By careful analysis, the total communication can be reduced to $O(\log n)$ bits.

Lower Bound: $D(R_{\mathbf{MAJ}}) = \Omega(\log n)$**.**

We prove this using a fooling set argument. Consider the set $\mathcal{F}$ of pairs $(x^{(j)}, y^{(j)})$ for $j = 1, 2, \ldots, n$, where:

$$x^{(j)} = 1^n \in X \quad \text{(all 1s)},$$
$$y^{(j)} = 0^{j-1}10^{n-j} \in Y \quad \text{(single 1 at position } j\text{)}.$$

Note that $|x^{(j)}| = n > n/2$ and $|y^{(j)}| = 1 \leq n/2$ (assuming $n \geq 2$), so $x^{(j)} \in X$ and $y^{(j)} \in Y$.

For each pair $(x^{(j)}, y^{(j)})$, the valid outputs are all indices $i \neq j$ (since $x_i^{(j)} = 1$ and $y_i^{(j)} = 0$ for $i \neq j$, while $x_j^{(j)} = y_j^{(j)} = 1$).

Now consider two distinct pairs $(x^{(j)}, y^{(j)})$ and $(x^{(k)}, y^{(k)})$ with $j \neq k$. The mixed pair $(x^{(j)}, y^{(k)})$ has $x_i^{(j)} \neq y_i^{(k)}$ for all $i \neq k$. In particular, index j is valid for $(x^{(j)}, y^{(k)})$ but not for $(x^{(j)}, y^{(j)})$.

By the fooling set argument, any deterministic protocol must distinguish between all n pairs in $\mathcal{F}$. Since the protocol produces a transcript that determines the output, and there are n distinct pairs requiring different behavior, there must be at least $\log_2 n$ bits of communication.

Therefore, $D(R_{\text{MAJ}}) = \Omega(\log n)$.

Combining the upper and lower bounds, we conclude that $D(R_{\text{MAJ}}) = \Theta(\log n)$. □

G.2.4 Gemini

<think>**Defining the Proof Strategy**

I'm currently focused on defining the core strategy for the proof. I started by analyzing the role and input – the theorem statement regarding the deterministic communication complexity of the majority relation. Initial thoughts center on identifying key components and establishing a framework for the proof's structure.

Refining the Approach

I'm now zeroing in on the Karchmer-Wigderson game aspect, recognizing the connection to the monotone depth of the Majority function. The relation R_{MAJ} directly maps to this concept. I'm considering how to leverage the Karchmer-Wigderson theorem and the properties of the Majority function to establish the communication complexity bound, specifically $D(R_{MAJ}) = \Theta(\log n)$. I'm also planning to use LaTeX code to express the proof.

Constructing the Proof Outline

My focus has shifted towards structuring the proof's core arguments. I'm leaning heavily on the Karchmer-Wigderson theorem and the known $O(\log n)$ circuit depth of the Majority function. I'm considering two main approaches: either directly constructing a communication protocol, or leveraging the existing circuit depth result to bound the communication complexity. The main challenge is to formalize these ideas into a rigorous proof using LaTeX code as per the instructions.

Formulating the Core Arguments

I'm now consolidating the proof's key steps. The circuit depth of $O(\log n)$ for the Majority function is my main instrument. I'm focusing on the Karchmer-Wigderson connection, and how the depth bounds the communication complexity. The protocol angle appears straightforward given the relation. The challenge is in formalizing the details and writing the LaTeX, ensuring a concise and rigorous presentation.

Outlining the Formal Proof

I'm now outlining a formal proof, ready for LaTeX implementation. My strategy involves leveraging the known circuit depth of the Majority function. The upper bound, $O(\log n)$, will be demonstrated by connecting the circuit depth to the communication complexity, directly using the Karchmer-Wigderson theorem. I'll need to show a lower bound of $\Omega(\log n)$ to complete the proof.

Implementing the LaTeX Proof

I'm translating the proof steps into a LaTeX format. The Karchmer-Wigderson theorem will be the core of the proof. I'm preparing to demonstrate the circuit depth of the Majority function using the carry-save adder approach, showing its $O(\log n)$ depth, hence establishing the upper bound. I'll need to detail the lower bound using a covering argument.

Structuring the Formal Proof

I'm presently building the proof framework, solidifying key arguments. The Karchmer-Wigderson connection is central, linking the relation to circuit depth. My plan is to show the upper bound through the depth of the Majority function. Then I will address the lower bound, constructing a proper "fooling set" argument, and illustrating the need for $\Omega(\log n)$ communication.

Establishing the Lower Bound

I'm now focusing on formalizing the lower bound argument within the LaTeX framework, specifically constructing the fooling set. I'm leveraging the pairs (x, y) where the only differing bit allows for output of any index. I'm preparing to show that the protocol must distinguish these, implying a $\Omega(\log n)$ lower bound.

Structuring the Formal Proof

I'm presently building the argument structure within the LaTeX framework, drawing heavily from the Karchmer-Wigderson connection. My plan includes an initial section to define the problem and relevant notations, followed by the upper bound proof, using the circuit depth of the Majority function. Then I'll focus on the lower bound, employing a carefully constructed "fooling set."

Synthesizing the Proof Outline

I'm now integrating all these ideas, ready to begin the LaTeX implementation. The central theme revolves around circuit depth and the Karchmer-Wigderson connection. I will demonstrate the upper bound via the Majority function's known circuit depth, and subsequently, I'll solidify the lower bound using the fooling set argument and the leaves of the protocol tree.

Refining the Lower Bound

I'm now focusing on formalizing the lower bound with the fooling set. I'm leveraging the pairs (x, y), where only the i-th bit differs. I'll construct a pair (x, y) such that $x \in X$, and $y \in Y$, where X and Y are specified by the relation. Therefore, all i's will be output, and so I can establish that the communication complexity of this protocol is $\Omega(\log n)$.

Formalizing Proof Structure

I'm presently building the proof within LaTeX. I'm leveraging the Karchmer-Wigderson connection for the upper bound. I have a detailed plan to show $MAJ_n \in NC^1$, and then address the lower bound with the fool set, focusing on pairs that differ in a single bit to prove a $\Omega(\log n)$ lower bound on the communication complexity.

Preparing LaTeX Implementation

I'm now writing LaTeX code to show the formal proof, leveraging the circuit depth argument and the Karchmer-Wigderson connection. I will start by showing the upper bound result using existing results for the Majority function circuit depth. Then, I will address the lower bound, emphasizing the fool set and resulting communication requirements.

Preparing the Proof Structure

I'm now fully prepared to write the LaTeX implementation of the proof. My strategy will leverage the depth of the Majority function circuit, and the Karchmer-Wigderson theorem. I'll include the necessary

definitions and notation to make the argument self-contained. Then, I'll demonstrate the upper and lower bounds to achieve the desired $\Theta(\log n)$ result, ensuring all the formal requirements are addressed.

</think>

Proof. We establish the upper and lower bounds separately to show that $D(R_{\text{MAJ}}) = \Theta(\log n)$.

Upper Bound. The relation R_{MAJ} corresponds to the Karchmer-Wigderson game for the Majority function, $f : \{0,1\}^n \to \{0,1\}$, defined by $f(z) = 1$ if $\sum_{i=1}^{n} z_i > n/2$ and $f(z) = 0$ otherwise. The Karchmer-Wigderson theorem states that the deterministic communication complexity $D(R_f)$ is equal to the minimum depth of a boolean circuit computing f over the basis $\{\wedge, \vee, \neg\}$. It is a standard result in circuit complexity that the Majority function is in the class NC^1, meaning it can be computed by a circuit of depth $O(\log n)$. Specifically, the arithmetic sum of n bits can be computed by a boolean circuit of depth $O(\log n)$ (for instance, using a wallace tree or carry-save adders to reduce the sum to two numbers, followed by a carry-lookahead adder). The comparison of this sum to the threshold $\lfloor n/2 \rfloor$ can also be performed by a circuit of depth $O(\log n)$. Since the circuit depth is $O(\log n)$, the communication complexity is bounded by

$$D(R_{\text{MAJ}}) \leq O(\log n).$$

Lower Bound. Let $\mathcal{P}$ be any deterministic protocol that solves R_{MAJ}. The protocol can be modeled as a binary tree where each internal node is owned by Alice or Bob and describes a bit to send, and each leaf is labeled with an output index $i \in \{1, \ldots, n\}$. For the protocol to be correct, for any input pair $(x, y) \in X \times Y$, if the execution of the protocol reaches a leaf labeled i, it must be the case that $x_i \neq y_i$.

We demonstrate that the set of leaves in the protocol tree must cover every index in $\{1, \ldots, n\}$. Let $j \in \{1, \ldots, n\}$ be an arbitrary index. We construct a specific pair of inputs $(x^{(j)}, y^{(j)})$ as follows. Let $k = \lfloor n/2 \rfloor$. We define $y^{(j)} \in \{0,1\}^n$ such that $y_j^{(j)} = 0$ and the Hamming weight of $y^{(j)}$ is exactly k. This is possible for any $n \geq 1$. Since $|y^{(j)}|_1 = k \leq |y^{(j)}|_0$, we have $y^{(j)} \in Y$. Next, we define $x^{(j)} \in \{0,1\}^n$ to be identical to $y^{(j)}$ at all positions $i \neq j$, and set $x_j^{(j)} = 1$. The Hamming weight of $x^{(j)}$ is $k + 1$. Since $k + 1 > n/2$, the number of 1s in $x^{(j)}$ is strictly greater than the number of 0s, implying $x^{(j)} \in X$.

Consider the pair $(x^{(j)}, y^{(j)})$. By construction, $x^{(j)}$ and $y^{(j)}$ differ only at the j-th coordinate. That is, $x_i^{(j)} = y_i^{(j)}$ for all $i \neq j$, and $x_j^{(j)} \neq y_j^{(j)}$. The definition of R_{MAJ} requires the protocol to output an index i such that $x_i^{(j)} \neq y_i^{(j)}$. The only such index is j. Therefore, on input $(x^{(j)}, y^{(j)})$, the protocol $\mathcal{P}$ must reach a leaf labeled j.

Since this holds for every $j \in \{1, \ldots, n\}$, the protocol tree must contain at least n distinct leaves. The depth of a binary tree with at least n leaves is at least $\lceil \log_2 n \rceil$. Thus,

$$D(R_{\text{MAJ}}) \geq \lceil \log_2 n \rceil = \Omega(\log n).$$

Combining the upper and lower bounds, we conclude that $D(R_{\text{MAJ}}) = \Theta(\log n)$. □

G.2.5 Grok

G.3 Problem 3

G.3.1 Problem Statement

Theorem G.3. *Prove that $C^D(R) \geq n^2$ is the best lower bound that can be proven by using Lemma 5.9.*

G.3.2 GPT

Proof. Let us consider the given inequality $C^D(R) \geq n^2$, where $C^D(R)$ is the quantity of interest, and we aim to prove that this is the best lower bound that can be proven using Lemma 5.9.

By the statement of Lemma 5.9, we know that there exists a certain bound on $C^D(R)$ depending on the properties of the function or system under consideration. Specifically, Lemma 5.9 provides a relationship that constrains the growth of $C^D(R)$ in terms of n. Without loss of generality, assume that the bound provided by Lemma 5.9 is of the form $C^D(R) \geq n^2$.

Now, we need to verify that this lower bound is the best possible. To do this, assume that there exists some $C^D(R) < n^2$ that satisfies the conditions of Lemma 5.9. If this were the case, then by the

assumptions of the lemma, it would imply a contradiction with the established relationships from the lemma. This contradiction arises because the form of the bound provided by Lemma 5.9 depends on the scaling of n, and the assumption $C^D(R) < n^2$ would violate this scaling.

Thus, we conclude that $C^D(R) \geq n^2$ is the strongest lower bound that can be established through the use of Lemma 5.9, and no smaller lower bound can be proven using the lemma.

Therefore, $C^D(R) \geq n^2$ is indeed the best lower bound that can be proven by Lemma 5.9. □

G.3.3 Claude

Proof. We prove that $C^D(R) \geq n^2$ is the best lower bound obtainable using Lemma 5.9 by showing two things: (1) Lemma 5.9 yields the bound $C^D(R) \geq n^2$, and (2) Lemma 5.9 cannot yield any better bound.

First, recall that Lemma 5.9 states that for any deterministic communication protocol computing a relation $R \subseteq X \times Y \times Z$, we have $C^D(R) \geq \log(\text{rank}(M_R))$, where M_R is the communication matrix associated with R and rank is computed over some field.

For the rectangle function $R : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ defined by $R(x,y) = 1$ if and only if $x_i = 1$ for all $i \leq k$ and $y_j = 1$ for all $j \leq \ell$ for some fixed k, ℓ, or more generally for the standard hard functions considered in communication complexity, the communication matrix M_R has dimensions $2^n \times 2^n$.

By Lemma 5.9, we have $C^D(R) \geq \log(\text{rank}(M_R))$. The maximum possible rank of an $2^n \times 2^n$ matrix is 2^n. Therefore, Lemma 5.9 gives us:

$$C^D(R) \geq \log(2^n) = n.$$

However, we need to establish the bound $C^D(R) \geq n^2$. This suggests we are considering a different setup where the input domain has size $N = 2^{n^2}$, so that n^2 bits are needed to specify each input. In this case, the communication matrix has dimensions $2^{n^2} \times 2^{n^2}$, and the maximum rank is 2^{n^2}.

Applying Lemma 5.9:

$$C^D(R) \geq \log(\text{rank}(M_R)) \leq \log(2^{n^2}) = n^2.$$

To show this is the best bound obtainable from Lemma 5.9, observe that Lemma 5.9 only uses the rank of the communication matrix. Since the rank of any $m \times m$ matrix is at most m, and our matrix has dimensions $2^{n^2} \times 2^{n^2}$, we have:

$$\text{rank}(M_R) \leq 2^{n^2}.$$

Therefore, Lemma 5.9 can yield at most:

$$C^D(R) \geq \log(2^{n^2}) = n^2.$$

Since Lemma 5.9 relies solely on the rank bound and the rank cannot exceed the dimension of the matrix, no better lower bound than n^2 can be proven using this lemma alone. Any improvement would require additional structural properties of R beyond what Lemma 5.9 captures through the rank measure. □

G.3.4 Gemini

<think>**Defining the Problem's Bounds**

I'm focused on grasping the specific parameters for Exercise 5.11. I am breaking down the core requirement, which is a rigorous mathematical proof. I'm taking the initial step of understanding the bounds, the variables, and the core operation in question: $C^D(R) \geq n^2$.

Unpacking the Core Concepts

I'm now carefully examining the relationship between Lemma 5.9 and Exercise 5.11. I've switched gears and am now focusing on the 'Tiles' problem context as hinted by the exercise. It's becoming clearer how Lemma 5.9 can potentially lead towards the desired lower bound, but requires further refinement of how it's specifically applied.

Connecting the Pieces

I'm now carefully revisiting the statement of Lemma 5.9 and its relation to the problem. After a review of the *Rectangle Size Bound* and the problem's setup, I can formulate a solid plan to tackle the exercise.

I'm focusing on the Clique vs. Independent Set problem. My goal is to use Lemma 5.9, or similar bounding techniques, to get the correct lower bound. I'm aiming for an initial proof outline, focusing on the connection between the input structure and the communication complexity measure.

Pinpointing the Lemma's Relevance

I'm now carefully analyzing the problem setup in relation to *Rectangle Size Bounds*. I've shifted my thinking to the core problem, namely, proving the lower bound using Lemma 5.9, as found in the specific context of the book *Communication Complexity*. I'm zeroing in on defining the relation and how Lemma 5.9's application will result in the n^2 lower bound for communication complexity.

Re-examining the Approach

I'm now revisiting the potential role of Lemma 5.9 and am trying to understand the exact structure of the relation R. After re-reading the relevant parts, I've confirmed that the exercise refers to the "best lower bound" derived from the rectangle size. I'm carefully trying to match this to an appropriate communication problem, possibly the shifting function, or a graph-related problem. I now think that n^2 relates to the number of leaves in the decision tree.

Refining the Approach

I'm now revisiting the concept of the "best lower bound" via Lemma 5.9, as found in the context of the *shifting function* or a closely-related communication problem. I'm focusing on how the rectangle size bound directly links to the n^2 complexity. My next step will be to define the relation and show the derivation.

Defining the Relation R

I'm now zeroing in on defining the precise relation R. Considering previous thoughts, I suspect that R may relate to a connectivity problem on a graph. The challenge is in defining a relation where Lemma 5.9, likely from a book by Kushilevitz and Nisan, would precisely yield an n^2 lower bound on communication complexity. My next move is testing this.

Revisiting Lemma and Bounds

I'm now carefully revisiting the statement of Lemma 5.9 and am trying to understand the exact structure of the relation R. The prompt is to prove that $C^D(R) \geq n^2$ is the best lower bound achievable using Lemma 5.9. I'm considering that the size of the relation R or the input size must be somehow proportional to n^2, perhaps as the number of edges in a graph with n vertices. The task is now to connect this to the specific techniques used for the lower bound.

Focusing on Lemma Definition

I'm now back to the fundamental concepts, trying to pinpoint the role of Lemma 5.9, specifically in relation to the $C^D(R) \geq n^2$ bound. I'm focusing on the Karchmer-Wigderson result. It seems that if we want $C^D(R) \geq n^2$, where R might be the relation of the Karchmer-Wigderson theorem, then R is about depth.

Focusing on Graph Problems

I'm now considering the communication complexity of graph problems. I'm focusing on why a lower bound of n^2 might be the best achievable using a Rectangle Size argument, as implied by Lemma 5.9. This suggests the number of rectangles is somehow related to the edges, and their total number is n^2.

Focusing on Graph Representation

I'm now contemplating representing the relation R using a graph, specifically with n vertices. The challenge is in defining the adjacency matrix and what each bit represents. I'm exploring the connectivity properties and how they might relate to the communication complexity of n^2.

Focusing on a Specific Relation

I'm now zeroing in on a specific relation to satisfy the $C^D(R) \geq n^2$ condition using Lemma 5.9. I'm considering scenarios where the size of the input dictates the complexity. It looks like the Karchmer-Wigderson result might be a good starting point, even though it focuses on depth. My next step will be to define this relation and its connection to the exercise.

Reconsidering Input Structure

I'm now revisiting the concept of input size relative to the claim of $C^D(R) \geq n^2$. Thinking of possible graph representations, particularly those with a number of edges as n^2, could lead me to the right relation to satisfy the communication complexity bound. I'm focusing on defining the relation.

Focusing on a Tight Bound

After extensive review, I've concluded the most likely approach involves using the Fooling Set argument, relating to the problem's setting. With a clear understanding of Lemma 5.9, and the context, I can now focus on how the problem may need an adjacency matrix for a graph problem with a complexity of n^2.

Analyzing Best Bound Implications

I'm now revisiting the concept of the "best" lower bound and its relation to Lemma 5.9, focusing on the $C^D(R) \geq n^2$ claim. I'm focusing on the input size required to achieve such a complexity, likely relating this to a graph representation where a fooling set argument might lead to n^2. I need to define a good relation for this problem.

Focusing on a Tight Bound

After extensive review, I've concluded the relation R is likely related to a *Fooling Set* argument. This will demonstrate that the input size, which might be the adjacency matrix of a graph, is closely related to the n^2 complexity. My next step will be to connect this argument.

Reiterating the Problem's Core

I am now focusing on solving Exercise 5.11, aiming to use Lemma 5.9, which I'm assuming refers to the Fooling Set argument, to demonstrate the n^2 lower bound. The challenge lies in defining the relation R, likely with an n^2 input size, that will satisfy the lower bound, while also emphasizing that this is the tightest bound achievable with the given lemma.

Reiterating the Core Task

I'm focused on tackling the core of Exercise 5.11, aiming to use Lemma 5.9, which I'm assuming refers to the Fooling Set argument, to get the n^2 lower bound. The input relation needs to be defined so I can prove the result.

Refining the Approach

I'm now revisiting the concept of input size relative to the bound and how the "best" lower bound can be proven using Lemma 5.9. Considering inputs that have an n^2 size, like an adjacency matrix, can result in the complexity. This approach focuses on proving the bound is tight.

Focusing on a Tight Bound

I'm now fully focused on creating a suitable relation R to apply the Fooling Set argument to Exercise 5.11, aiming to use Lemma 5.9, and ultimately proving the tight n^2 lower bound. I'm focusing on why we cannot achieve a greater lower bound using this argument.

Focusing on a Specific Example

I'm now zeroing in on an appropriate choice for R. Given the "best lower bound" statement, the relation's structure needs careful consideration, likely relating to a function on a matrix where a fooling set argument will give n^2. I'm thinking of the identity function on n^2 bits.

Reiterating Proof Structure

I'm now completely focused on formulating the final LaTeX proof. I have a firm plan based on the Fooling Set argument, which, I am assuming, is equivalent to Lemma 5.9. I'm focusing on defining the relation, and applying this to satisfy the problem requirement.

Formulating the Final Proof

I am now completely focused on crafting the final proof, solidifying my assumptions about Lemma 5.9, which I believe is the *Fooling Set* argument. My strategy centers on defining an appropriate relation and demonstrating how the lemma provides the precise n^2 lower bound, thus proving it is the tightest possible.

Formulating the Proof Structure

I'm now fully immersed in crafting the final proof, leveraging my understanding of the Fooling Set argument as the basis for Lemma 5.9. The problem centers around demonstrating the tight n^2 lower bound. I'm focusing on creating an effective relation R and showing the limitations of the Lemma.

Formalizing the Approach

I'm now carefully building the structure of my formal proof. I'm focusing on the Fooling Set argument and how to best explain it for Exercise 5.11, as the argument provides the *best* lower bound. I'm taking the previous steps and turning them into mathematical notation.

Revisiting Proof Construction

I'm now carefully revisiting the final proof structure, assuming the Fooling Set Lemma (Lemma 5.9) and a relation on $N = n^2$ bits. The goal is to prove n^2 is the best lower bound achievable using this lemma. I'm reviewing the Identity function approach for the proof.

Formalizing the Proof's Core

I'm now fully focused on creating a suitable relation R to apply the Fooling Set argument to Exercise 5.11, aiming to use Lemma 5.9. The problem's core is to demonstrate the tight n^2 lower bound. I'm focusing on defining a relation and its related argument.

</think>

Proof. We assume the context of deterministic communication complexity where the inputs x and y represent structures of size $N = n^2$ (e.g., adjacency matrices of graphs on n vertices). We interpret Lemma 5.9 as the standard combinatorial lower bound technique for deterministic communication complexity, specifically the Fooling Set Lemma (or equivalently, the logarithm of the size of a monochromatic rectangle cover).

Lemma 5.9 states that for a relation $R \subseteq X \times Y \times Z$, if there exists a fooling set $S \subseteq X \times Y$ of size t, then:

$$C^D(R) \geq \log_2 t.$$

To prove that n^2 is the best lower bound that can be proven using this lemma, we must show two things: 1. The lemma cannot yield a lower bound strictly greater than n^2. 2. There exists a relation R for which the lemma yields exactly the lower bound n^2.

Part 1: Upper bound on the lemma's estimate. Let $X = Y = \{0,1\}^{n^2}$. A fooling set S is a set of input pairs $\{(x_i, y_i)\}_{i=1}^{t}$ with specific properties regarding the function values. A necessary condition for a set to be a fooling set for any function (or relation) is that the elements must be distinguishable in a way that prevents them from forming a monochromatic rectangle. Specifically, for a deterministic protocol, the projection of a fooling set onto X (or Y) must be injective. If two pairs (x, y) and (x, y') share the same x, they can only belong to the same fooling set if the function values satisfy specific constraints, but in the standard fooling set construction for lower bounds (e.g., for the equality or disjointness functions), the elements x in the pairs (x, x) are distinct.

More formally, the size of any fooling set S is bounded by the size of the input space. Since $S \subseteq X \times Y$ and the associated rectangle cover corresponds to disjoint inputs or specific diagonal structures, the size of the fooling set t satisfies:

$$t \leq |X| = 2^{n^2}.$$

Applying Lemma 5.9, the lower bound obtained is:

$$\text{Lower Bound} = \log_2 t \leq \log_2(2^{n^2}) = n^2.$$

Thus, Lemma 5.9 cannot prove a lower bound strictly greater than n^2.

Part 2: Achievability of the bound. Consider the Identity (or Equality) relation EQ on n^2 bits, defined by $EQ(x, y) = 1$ if $x = y$ and 0 otherwise, where $x, y \in \{0,1\}^{n^2}$. Construct the set $S = \{(x, x) \mid x \in \{0,1\}^{n^2}\}$. For any two distinct pairs $(x, x), (x', x') \in S$: 1. $EQ(x, x) = EQ(x', x') = 1$. 2. $EQ(x, x') = 0$ (since $x \neq x'$). 3. $EQ(x', x) = 0$. This satisfies the definition of a fooling set. The size of this set is:

$$|S| = |\{0,1\}^{n^2}| = 2^{n^2}.$$

Applying Lemma 5.9 yields:

$$C^D(EQ) \geq \log_2(2^{n^2}) = n^2.$$

Since the trivial protocol for EQ requires Alice to send her entire n^2-bit input to Bob, $C^D(EQ) \leq n^2$ (ignoring constant overhead). Thus, the bound is tight.

Conclusion: Since the method of Lemma 5.9 is bounded from above by n^2 for inputs of length n^2, and there exists a relation achieving this bound, $C^D(R) \geq n^2$ is the best lower bound that can be proven using Lemma 5.9. □

G.3.5 Grok

Proof. By Lemma 5.9, the deterministic cover number $C^D(R)$ of a relation R satisfies

$$C^D(R) \geq \frac{|R|}{\max_z |\{(x,y) \mid (x,y,z) \in R\}|},$$

where $|R|$ denotes the number of triples in R.

For the given relation R, there are $|R| = \Theta(n^2 \cdot 2^{2n})$ triples, since there are n^2 possible witnesses $z = (i,j)$ and for each fixed (i,j) there are $\Theta(2^{2n})$ pairs (x,y) such that $(x,y,(i,j)) \in R$.

The maximum fiber size is $\max_{(i,j)} |\{(x,y) \mid (x,y,(i,j)) \in R\}| = \Theta(2^{2n})$.

Thus,

$$C^D(R) \geq \frac{\Theta(n^2 \cdot 2^{2n})}{\Theta(2^{2n})} = \Theta(n^2).$$

To show that this bound is tight (and hence the strongest provable using Lemma 5.9), consider the n^2 rectangles defined for each $(i,j) \in [n] \times [n]$ as the set of all (x,y) such that the fixed witness (i,j) is valid. Each such rectangle is monochromatic for (i,j), and these n^2 rectangles together cover all triples in R (as the fibers for different (i,j) are disjoint in terms of triples). Therefore, $C^D(R) \leq n^2$.

It follows that $C^D(R) = \Theta(n^2)$, so the lower bound of $\Omega(n^2)$ obtained from Lemma 5.9 is tight, and no stronger asymptotic lower bound on $C^D(R)$ can be proven using this lemma. □

G.4 Problem 4

G.4.1 Problem Statement

Theorem G.4. *Prove that* $D(\mathit{FORK}) = O(\log \ell \log w)$.

G.4.2 GPT

Proof. We are tasked with proving that $D(\text{FORK}) = O(\log \ell \log w)$.

Let us first define the parameters involved. Let $D(\text{FORK})$ represent the complexity of the FORK operation. Here, ℓ denotes a parameter related to the size of the input, and w represents a parameter related to the width or range of the problem.

We begin by analyzing the structure of the FORK operation. The FORK operation can be viewed as a process that involves recursive division or branching, typically leading to a logarithmic depth of recursion. In the case of branching processes, the depth of recursion is directly related to the logarithm of the size of the problem, which in this context is represented by ℓ.

For each level of recursion, the work required to process the elements at that level can be influenced by the width w of the problem. This suggests that the total work done at each level scales logarithmically with respect to both ℓ and w.

Thus, at each recursive level, the work is proportional to $O(\log \ell \log w)$. Since the depth of recursion is also logarithmic in ℓ, we obtain a total complexity of the form $O(\log \ell \log w)$ for the FORK operation.

To formalize this further, suppose that at each level of recursion, the work required is bounded by $O(\log \ell \log w)$. Since the depth of recursion is $O(\log \ell)$, the total work done by the algorithm is the product of the work at each level and the number of levels. This gives the overall complexity as:

$$D(\text{FORK}) = O(\log \ell \log w)$$

Thus, we have proven that $D(\text{FORK}) = O(\log \ell \log w)$, as required. □

G.4.3 Claude

Proof. We prove that the depth of the FORK circuit satisfies $D(\text{FORK}) = O(\log \ell \log w)$, where ℓ denotes the length of the input and w denotes the word size.

The FORK circuit computes a function that distributes an input to multiple outputs. The circuit can be constructed using a binary tree structure where each node performs a simple routing operation.

First, we establish the structure of the FORK circuit. The circuit takes an input of length ℓ and produces multiple copies or routes the input to w different destinations. This can be implemented as a complete binary tree with w leaves.

The depth of a complete binary tree with w leaves is $\lceil \log_2 w \rceil = O(\log w)$. Each node in this tree performs operations on words of size ℓ bits.

At each level of the tree, we need to process the ℓ-bit input. The processing at each node involves copying or routing the input, which can be done using basic gates. For an ℓ-bit word, the depth required to perform the necessary operations at each node is $O(\log \ell)$ when using efficient parallel circuits for word-level operations.

More precisely, each node in the FORK tree must handle an ℓ-bit word and route it to its children. Using standard techniques from parallel computation, operations on ℓ-bit words can be performed in depth $O(\log \ell)$ using circuits of polynomial size. This includes operations such as copying, multiplexing, or conditional routing.

Since the tree has depth $O(\log w)$ and each level requires depth $O(\log \ell)$ for processing ℓ-bit words, the total depth of the FORK circuit is bounded by:

$$D(\text{FORK}) = O(\log w) \cdot O(\log \ell) = O(\log \ell \log w).$$

To verify this bound is tight, observe that:

(i) We cannot reduce the tree depth below $\Omega(\log w)$ since we must distinguish between w different outputs, requiring at least $\log w$ bits of information to be processed sequentially in the worst case.

(ii) We cannot process ℓ-bit words in depth less than $\Omega(\log \ell)$ for general word operations in the standard circuit model.

Therefore, we conclude that $D(\text{FORK}) = O(\log \ell \log w)$. □

G.4.4 Gemini

Proof. Let $D(\ell, w)$ denote the depth (span) of the FORK algorithm on an input of length ℓ and width w. We analyze the complexity based on the standard divide-and-conquer paradigm applicable to this class of problems, typically involving the composition of transition functions or path computations in a layered structure of width w.

The algorithm proceeds by partitioning the input sequence of length ℓ into two adjacent subsegments, each of length $\ell/2$. These subproblems are solved recursively. Since the two subproblems are independent, they are executed in parallel. Consequently, the depth contribution from the recursive step is $\max(D(\ell/2, w), D(\ell/2, w)) = D(\ell/2, w)$.

Following the parallel execution of the subproblems, a merge step is required to combine the partial results. In the context of width-w dependencies, the partial results are represented by transition structures (e.g., matrices or functional operators) of size $w \times w$. The combination of two such structures corresponds to an associative composition operation, such as matrix multiplication over a semiring. The depth of composing two $w \times w$ structures in parallel is bounded by the depth of a reduction tree over the dimension w, which is $O(\log w)$.

Therefore, the depth $D(\ell, w)$ satisfies the recurrence relation:

$$D(\ell, w) = D(\ell/2, w) + c \log w,$$

for some constant $c > 0$, with the base case $D(1, w) = O(\log w)$ (or $O(1)$, which does not alter the asymptotic bound).

We solve this recurrence by summation over the levels of the recursion tree. The recursion depth is $\log_2 \ell$. At each level k for $0 \leq k < \log_2 \ell$, the non-recursive work contributes $O(\log w)$ to the critical path. Summing these contributions yields:

$$D(\ell, w) = \sum_{k=0}^{\log_2 \ell - 1} c \log w = c \log w \cdot \log_2 \ell.$$

By properties of logarithms, this is equivalent to:

$$D(\ell, w) = O(\log \ell \log w).$$

Thus, we conclude that $D(\text{FORK}) = O(\log \ell \log w)$. □

G.4.5 Grok

Proof. The relation FORK is defined over inputs $x, y \in [w]^\ell$ with the promise that there exists at least one $i \in [\ell]$ such that $x_j = y_j$ for all $j < i$ and $x_j \neq y_j$ for all $j \geq i$, and $(x, y, i) \in \text{FORK}$ precisely when this condition holds for that i (typically unique under the promise that disagreements form a suffix).

We describe a deterministic protocol that outputs a valid i using $O(\log \ell \log w)$ bits of communication.

The protocol maintains an interval $[L, R] \subseteq [\ell]$, initially $L = 1$ and $R = \ell$, with the invariant that any valid fork position lies in $[L, R]$.

While $L < R$:

Both parties compute $\text{mid} = \lfloor (L + R)/2 \rfloor$.

Alice sends the value $x_{\text{mid}} \in [w]$ to Bob, using $O(\log w)$ bits.

Bob receives x_{mid} and compares it to y_{mid}.

- If $x_{\text{mid}} = y_{\text{mid}}$, then any valid fork position satisfies $i > \text{mid}$ (since if $i \leq \text{mid}$, then $\text{mid} \geq i$ implies $x_{\text{mid}} \neq y_{\text{mid}}$).

Bob sends 1 bit indicating "equal".

Both set $L := \text{mid} + 1$.

- If $x_{\text{mid}} \neq y_{\text{mid}}$, then any valid fork position satisfies $i \leq \text{mid}$ (since if $i > \text{mid}$, then $\text{mid} < i$ implies $x_{\text{mid}} = y_{\text{mid}}$).

Bob sends 1 bit indicating "differ".

Both set $R := \text{mid}$.

The invariant is preserved in both cases, as the promise ensures the disagreements form a suffix starting at the true fork position.

Each iteration halves the interval length (up to constant factors), so there are $O(\log \ell)$ iterations.

Per iteration, $O(\log w)$ bits are sent by Alice and $O(1)$ bit by Bob.

When $L = R$, both parties output $i = L$, which is a valid fork position by the invariant and the promise.

The total communication is thus $O(\log \ell \log w)$.

Therefore, $D(\text{FORK}) = O(\log \ell \log w)$. □

G.5 Problem 5

G.5.1 Problem Statement

Theorem G.5. *Let* FORK′ *be the relation consisting of all triples* (x, y, i) *such that* $x, y \in \Sigma^\ell$ *and* i *is such that* $x_i = y_i$ *and either* $x_{i+1} \neq y_{i+1}$ *or* $x_{i-1} \neq y_{i-1}$. *Then* $D(\text{FORK}') = \Omega(\log \ell \log w)$.

G.5.2 GPT

Proof. We are given the relation FORK′ consisting of all triples (x, y, i) such that $x, y \in \Sigma^\ell$ and i is such that $x_i = y_i$ and either $x_{i+1} \neq y_{i+1}$ or $x_{i-1} \neq y_{i-1}$. We are tasked with proving that $D(\text{FORK}') = \Omega(\log \ell \log w)$.

Let $x, y \in \Sigma^\ell$, where ℓ is the length of the strings and Σ is the alphabet. We define the relation FORK′ as follows: for a triple (x, y, i) to belong to FORK′, the following conditions must hold: 1. $x_i = y_i$ for some $i \in \{1, 2, \ldots, \ell\}$, 2. Either $x_{i+1} \neq y_{i+1}$ or $x_{i-1} \neq y_{i-1}$, assuming the indices are valid.

Now, we aim to show that the defect $D(\text{FORK}')$, which is the minimum number of probes required to distinguish all possible pairs (x, y) such that $(x, y, i) \in \text{FORK}'$ for some i, is lower-bounded by $\Omega(\log \ell \log w)$, where w is the size of the alphabet Σ.

Step 1: Understanding the structure of FORK′ Consider the set of strings $x, y \in \Sigma^\ell$. The relation FORK′ holds when there is an index i such that $x_i = y_i$ and there is a discrepancy in at least one of the neighboring positions: either $x_{i+1} \neq y_{i+1}$ or $x_{i-1} \neq y_{i-1}$. This means that the relation FORK′ is defined by a small neighborhood around i, specifically by the values of x_{i-1}, x_i, and x_{i+1}.

Thus, for each valid i, the relation can be seen as partitioning the space of possible strings x and y into distinct blocks, where two strings are considered related if they agree on a specific index i and differ in the neighborhood of that index.

Step 2: Lower bound on the number of distinct triples (x, y, i) There are ℓ possible values for the index i, and for each i, there are w^2 possible combinations of the neighboring symbols (x_{i-1}, x_{i+1}) that differ between x and y. Thus, the total number of distinct pairs (x, y) related by FORK$'$ is at least $\ell \cdot w^2$.

Step 3: Bounding the defect $D(\text{FORK}')$ The defect $D(\text{FORK}')$ represents the minimum number of probes needed to distinguish between all pairs (x, y) that are related by FORK$'$. Since there are at least $\ell \cdot w^2$ such pairs, and each probe provides some information about the values of x_i or x_{i-1} or x_{i+1}, the number of probes required to distinguish all such pairs is at least the logarithm of the total number of distinct pairs. Therefore, we have:

$$D(\text{FORK}') \geq \log(\ell \cdot w^2) = \log \ell + 2 \log w$$

Since $\log w$ is at least constant for any fixed alphabet size w, the term $2 \log w$ can be absorbed into the constant factor. Thus, we have:

$$D(\text{FORK}') = \Omega(\log \ell)$$

Step 4: Final bound Combining the above, we see that the defect is lower-bounded by $\Omega(\log \ell \log w)$, which completes the proof.

□

G.5.3 Claude

Proof. We establish a lower bound on the deterministic communication complexity of FORK$'$ by reduction from the Index problem.

Let ℓ and w be parameters where ℓ represents the length of strings in Σ^ℓ and $w = |\Sigma|$ is the alphabet size. We proceed by showing that any deterministic protocol computing FORK$'$ requires $\Omega(\log \ell \log w)$ bits of communication.

First, we reduce from the Index problem, where Alice holds a string $s \in \Sigma^n$ and Bob holds an index $j \in [n]$, and they must determine s_j. It is well-known that $D(\text{Index}) = \Omega(n \log w)$ for strings over an alphabet of size w.

We construct the reduction as follows. Given an instance (s, j) of Index with $s \in \Sigma^n$ and $j \in [n]$, Alice constructs a string $x \in \Sigma^\ell$ where $\ell = 2n + 1$ by setting:

$$x_i = \begin{cases} s_{\lceil i/2 \rceil} & \text{if } i \text{ is odd} \\ \sigma_0 & \text{if } i \text{ is even} \end{cases}$$

where $\sigma_0 \in \Sigma$ is a fixed symbol.

Bob constructs $y \in \Sigma^\ell$ by setting $i = 2j$ and:

$$y_k = \begin{cases} x_k & \text{if } k \neq i - 1 \\ \tau & \text{if } k = i - 1 \end{cases}$$

where $\tau \in \Sigma$ is chosen uniformly at random from $\Sigma \setminus \{\sigma_0\}$.

Observe that $x_i = y_i = \sigma_0$ and $x_{i-1} = s_j$ while $y_{i-1} = \tau \neq \sigma_0$. Thus $x_{i-1} \neq y_{i-1}$ if and only if $s_j \neq \tau$.

By querying FORK$'$ on (x, y, i), we can determine whether the triple is in the relation. Since the triple $(x, y, i) \in$ FORK$'$ holds, and we know $x_{i-1} \neq y_{i-1}$ occurs when $s_j \neq \tau$, by trying all possible values of τ (at most $w - 1$ attempts), Bob can determine s_j.

This gives us a protocol for Index using $O(w)$ calls to FORK$'$. Therefore:

$$D(\text{Index}) \leq O(w) \cdot D(\text{FORK}')$$

Since $D(\text{Index}) = \Omega(n \log w)$ and $n = \Theta(\ell)$, we have:

$$\Omega(\ell \log w) \leq O(w) \cdot D(\text{FORK}')$$

This yields $D(\text{FORK}') = \Omega(\frac{\ell \log w}{w})$.

To obtain the stronger bound of $\Omega(\log \ell \log w)$, we use a more refined reduction. We embed a smaller Index instance by using only $\log \ell$ positions in the strings. Specifically, Alice and Bob agree on a subset $S \subseteq [\ell]$ with $|S| = \Theta(\log \ell)$, and encode an Index instance of length $\Theta(\log \ell)$ into these positions.

Following the same reduction strategy on this subset, we obtain:

$$D(\text{FORK}') = \Omega(\log \ell \log w)$$

This completes the proof. □

G.5.4 Gemini

Proof. We prove the lower bound $D(\text{FORK}') = \Omega(\log \ell \log w)$ by reducing the problem of finding the first index of difference between two strings to FORK′. Let $\mathcal{P}$ be the problem where Alice holds $x \in \Sigma^\ell$ and Bob holds $y \in \Sigma^\ell$, with the promise that there exists a unique index $k \in \{1, \ldots, \ell\}$ such that $x_i = y_i$ for all $i < k$ and $x_k \neq y_k$. We aim to find the index $k - 1$ (or 0 if $k = 1$). Note that finding such a k allows us to identify a valid fork at $i = k - 1$ (assuming $x_0 = y_0$ is fixed or the condition is adapted for boundary). Thus, $D(\text{FORK}') \geq D(\mathcal{P})$.

We employ an adversary argument to lower bound $D(\mathcal{P})$. Let the current search space for the index k be an interval $I \subseteq \{1, \ldots, \ell\}$. Initially, $I = [1, \ell]$. The adversary maintains a set of valid inputs consistent with the communication transcript so far. Specifically, the adversary maintains that for any $k \in I$, there exists a pair of inputs (x, y) consistent with the transcript such that their first difference is at index k.

Consider the state where the search interval is $I = [a, b]$. To reduce the size of I by a constant factor (e.g., to distinguish between $k \in [a, m]$ and $k \in [m + 1, b]$ where $m = (a + b)/2$), the players must determine whether x and y differ in the interval $[a, m]$. Given the promise that x and y agree on $1, \ldots, a - 1$, checking if the first difference lies in $[a, m]$ is equivalent to checking if $x_{[a,m]} = y_{[a,m]}$. Since the adversary can fix $x_i = y_i$ for all $i \in [a, m - 1]$ without loss of generality (pushing the difference to m), the bottleneck reduces to checking equality at the critical index m, i.e., determining if $x_m = y_m$.

The communication complexity of testing equality for two symbols $u, v \in \Sigma$ is $\Theta(\log |\Sigma|) = \Theta(\log w)$. The adversary can ensure that the players cannot reliably determine if $x_m = y_m$ (and thus whether $k \leq m$ or $k > m$) with fewer than $\Omega(\log w)$ bits of communication. Specifically, for any protocol transmitting $c < \log w$ bits, the partition of the input space induced by the transcript is consistent with both $x_m = y_m$ and $x_m \neq y_m$ for some choices of symbols, preventing the players from eliminating either sub-interval.

Therefore, to reduce the size of the candidate interval I by a factor of 2, the protocol must expend $\Omega(\log w)$ bits. Since the initial interval size is ℓ, the number of such reduction steps required is $\Omega(\log \ell)$. The total communication complexity is thus:

$$D(\text{FORK}') \geq D(\mathcal{P}) = \Omega(\log \ell \cdot \log w).$$

□

G.5.5 Grok

Proof. The relation FORK′ consists of triples $(x, y, i) \in \Sigma^\ell \times \Sigma^\ell \times [\ell]$ where $|\Sigma| = w$, $x_i = y_i$, and $x_{i-1} \neq y_{i-1}$ or $x_{i+1} \neq y_{i+1}$ (with conventions that nonexistent positions contribute no constraint to the disjunction).

To prove $D(\text{FORK}') = \Omega(\log \ell \log w)$, we use the technique of partially correct protocols.

A deterministic protocol Π is a c-bit (α, ℓ)-protocol if there exists a set $S \subseteq \Sigma^\ell$ with $|S| \geq \alpha \cdot w^\ell$ such that for every $x, y \in S$, Π on input (x, y) outputs some i with $(x, y, i) \in \text{FORK}'$.

Let $C(\alpha, \ell)$ be the minimal c such that there exists a c-bit (α, ℓ)-protocol. Then $D(\text{FORK}') = C(1, \ell)$.

We show $C(w^{-O(1)}, \ell) = \Omega(\log \ell \log w)$.

First, a round-elimination lemma: If there is a c-bit (α, ℓ)-protocol, then there is a $(c-1)$-bit $(\alpha/2, \ell)$-protocol.

Assume Alice sends the first bit $b \in \{0,1\}$. Let $S = S_0 \cup S_1$ be partitioned by the bit sent on inputs in S. Let S' be the larger part; $|S'| \geq \alpha w^\ell/2$. The protocol omitting the first bit and hardcoding b is correct on $S' \times S'$ and uses $c-1$ bits.

Iterating $\Omega(\log w)$ times yields $C(w^{-O(1)}, \ell) = \Omega(\log w) + C(w^{-O(1)}, \ell)$.

Next, an amplification lemma: If $\alpha \geq \lambda/w$ for large constant λ and there is a c-bit (α, ℓ)-protocol, then there is a c-bit $(O(\sqrt{\alpha}), \ell/2)$-protocol.

Let S witness the (α, ℓ)-protocol. Split strings into prefix of length $\ell/2$ and suffix of length $\ell/2$. Consider the bipartite graph with left vertices prefixes, right suffixes, edge if prefix $\circ$ suffix $\in S$. The density is at least α.

By averaging, either some prefix u has degree at least $\sqrt{\alpha} \cdot w^{\ell/2}$, or many prefixes have degree at least $\alpha/2 \cdot w^{\ell/2}$.

In the first case, fix common prefix u for both parties. Then the left half is identical. Disagreements can only occur in the right half. Valid i must satisfy the local condition using disagreements in the right. The protocol on suffixes, simulating with fixed u, outputs a valid i shifted to the right half, corresponding to a valid witness for the $\ell/2$ instance on suffixes. The good set for suffixes has size at least $\sqrt{\alpha} \cdot w^{\ell/2}$.

In the second case, there are many good prefixes. By probabilistic method (similar to the asymmetric case), find complementary suffixes that differ in every position in the right half while keeping the full string in S. Then, the right half always disagrees everywhere, so no agreements in right half. Thus valid i must be in left half. The output i corresponds directly to a valid witness for the $\ell/2$ instance on prefixes.

Combining as in the recurrence: starting with small $\alpha = w^{-O(1)}$, each halving of ℓ costs $O(\log w)$ bits by repeated round elimination before amplifying. Over $\log \ell$ halvings, the total is $\Omega(\log \ell \log w)$.

Thus $D(\text{FORK}') = \Omega(\log \ell \log w)$. □

H Problems in Chapter 6

H.1 Problem 1

H.1.1 Problem Statement

Theorem H.1 (Exercise 6.1)**.** *Let $x_1, \ldots, x_k$ each be an n-bit string. Define the generalized equality function $\mathrm{EQ}_n^k(x_1, \ldots, x_k)$ to be 1 if all k strings are equal, and the generalized nonequality function $\mathrm{NE}_n^k(x_1, \ldots, x_k)$ to be 1 if all k strings are distinct (EQ_n^k and NE_n^k are complements only for $k = 2$). Use reductions from the two-party model to show that if player i knows only x_i, then the communication complexity of EQ_n^k is $\Theta(n)$ and the communication complexity of NE_n^k is $\Theta(kn)$.*

H.1.2 GPT

Proof. We will prove the communication complexities of the generalized equality function EQ_n^k and the generalized nonequality function NE_n^k through reductions from the two-party communication model.

Communication Complexity of EQ_n^k:

The generalized equality function $\mathrm{EQ}_n^k(x_1, \ldots, x_k)$ outputs 1 if all the strings $x_1, \ldots, x_k$ are equal, and 0 otherwise. Suppose that player i knows only x_i. The task is to determine the communication complexity of EQ_n^k when the players are allowed to exchange messages.

Consider the following reduction: we reduce the communication complexity of EQ_n^k to the problem of equality testing between two strings of length n. The classical equality function $\mathrm{EQ}_2(x_1, x_2)$ outputs 1 if $x_1 = x_2$, and 0 otherwise. We will show that solving EQ_n^k requires at least $\Theta(n)$ bits of communication.

First, observe that the problem of determining whether all k strings $x_1, \ldots, x_k$ are equal can be reduced to the classical EQ_2 problem as follows: one player sends a message to the other containing the XOR of all k strings. Specifically, one player computes the XOR of $x_1, \ldots, x_k$ and sends this message. The other player can then check if the XOR is zero. If the XOR is zero, all the strings are equal; otherwise, they are not. This reduction shows that the communication complexity of EQ_n^k is at least $\Theta(n)$, because the communication complexity of EQ_2 is $\Theta(n)$.

Since EQ_n^k requires at least $\Theta(n)$ bits to determine equality, it follows that the communication complexity of EQ_n^k is $\Theta(n)$.

Communication Complexity of NE_n^k:

Next, we consider the generalized nonequality function $\mathrm{NE}_n^k(x_1, \ldots, x_k)$. This function outputs 1 if all k strings are distinct, and 0 otherwise. We aim to show that the communication complexity of NE_n^k is $\Theta(kn)$.

The key idea is that in order to determine whether all k strings are distinct, each player must communicate information about their string that allows the other players to check for equality among the strings. The communication complexity of NE_n^k can be reduced to the problem of checking distinctness among k strings.

We can reduce NE_n^k to a series of equality tests. Suppose we have k players, each with an n-bit string. To determine if all the strings are distinct, each player sends their string to all other players. If any two strings are the same, the function NE_n^k outputs 0; otherwise, it outputs 1. Since each player must send their string to all the other players, the total number of bits communicated is $k \cdot n$, resulting in a communication complexity of $\Theta(kn)$.

Therefore, the communication complexity of NE_n^k is $\Theta(kn)$.

□

H.1.3 Claude

Proof. We prove the communication complexity bounds for both functions using reductions from the two-party communication model.

Part 1: Communication complexity of EQ_n^k is $\Theta(n)$.

Lower bound ($\Omega(n)$): We reduce the standard two-party equality function EQ_n^2 to EQ_n^k. Suppose players 1 and 2 wish to compute $\mathrm{EQ}_n^2(y_1, y_2)$ where player 1 knows y_1 and player 2 knows y_2. We construct a protocol for EQ_n^k as follows: set $x_1 = y_1$, $x_2 = y_2$, and $x_3 = \cdots = x_k = y_1$ (where players $3, \ldots, k$ receive y_1 from player 1). Then $\mathrm{EQ}_n^k(x_1, \ldots, x_k) = 1$ if and only if $y_1 = y_2 = x_3 = \cdots = x_k$, which holds if and only if $y_1 = y_2$. Thus $\mathrm{EQ}_n^k(x_1, \ldots, x_k) = \mathrm{EQ}_n^2(y_1, y_2)$. Since the deterministic communication complexity of EQ_n^2 is $n + 1$ (one player must send their entire string), and any protocol for EQ_n^k yields a protocol for EQ_n^2, we have $\mathrm{CC}(\mathrm{EQ}_n^k) = \Omega(n)$.

Upper bound ($O(n)$): Player 1 broadcasts x_1 to all other players using n bits. Each player i (for $i = 2, \ldots, k$) locally checks if $x_i = x_1$ and broadcasts 1 bit indicating equality or inequality. If all players broadcast agreement, output 1; otherwise output 0. The total communication is $n + (k - 1) = O(n)$ since k is constant or absorbed into the asymptotic notation when k depends on n polynomially.

Therefore, $\mathrm{CC}(\mathrm{EQ}_n^k) = \Theta(n)$.

Part 2: Communication complexity of NE_n^k is $\Theta(kn)$.

Lower bound ($\Omega(kn)$): We use a reduction argument based on the index function. Consider k instances of two-party equality problems. For each pair (i, j) with $1 \leq i < j \leq k$, we need to verify that $x_i \neq x_j$. There are $\binom{k}{2} = \Theta(k^2)$ such pairs. However, we use a more direct argument: we reduce from the problem where players must distinguish between two cases: (1) all strings are distinct, or (2) at least two strings are equal.

Consider the following reduction: suppose we have k two-party equality problems, where in the i-th problem, players must determine if strings y_i and z_i are equal. We can embed this into NE_n^k by setting x_i to encode information about multiple y_j values. By a counting argument, the number of distinct n-bit strings that can be communicated is at most 2^c where c is the communication complexity. To distinguish all possible configurations of k distinct strings from $k!$ permutations of k specific strings requires $\Omega(k \log k)$ bits. However, a more refined argument using the fooling set method shows that we need $\Omega(kn)$ bits: consider inputs where each x_i differs from all others in a specific subset of coordinates. The fooling set has size $2^{kn/k} = 2^n$ for each player, giving $\Omega(kn)$ communication.

More directly, player i must communicate enough information about x_i so that all other players can verify distinctness. Since there are k players and each must communicate $\Omega(n)$ bits to ensure their string is distinguished from all others, the total is $\Omega(kn)$.

Upper bound ($O(kn)$): Each player i broadcasts their entire string x_i using n bits. After all broadcasts, every player knows all k strings and can locally verify if they are all distinct. The total communication is kn bits.

Therefore, $\mathrm{CC}(\mathrm{NE}_n^k) = \Theta(kn)$. □

H.1.4 Gemini

Proof. We analyze the deterministic communication complexity of the generalized equality and nonequality functions using reductions from the two-party model.

First, we consider $\mathrm{EQ}_n^k(x_1, \ldots, x_k)$. For the upper bound, we define a protocol where Player 1 broadcasts their input x_1 to all other players. This requires n bits of communication. Subsequently, every other Player i (for $i = 2, \ldots, k$) compares their local input x_i with the received x_1 and broadcasts a single bit b_i, where $b_i = 1$ if $x_i = x_1$ and 0 otherwise. The function evaluates to 1 if and only if $\bigwedge_{i=2}^k b_i = 1$. The total communication cost is $n + k - 1$. Assuming $k = O(n)$, this is $O(n)$. For the lower bound, we reduce the two-party equality problem EQ_n^2 to EQ_n^k. Let Alice and Bob hold inputs $x, y \in \{0,1\}^n$. They partition the k players such that Alice simulates Player 1 and Players $3, \ldots, k$, while Bob simulates Player 2. Alice assigns $x_1 = x$ and $x_i = x$ for $i \geq 3$. Bob assigns $x_2 = y$. The generalized equality $\mathrm{EQ}_n^k(x, y, x, \ldots, x)$ is 1 if and only if $x = y$ (since $x = x$ is always true). Any protocol for EQ_n^k can thus solve EQ_n^2. Since the deterministic communication complexity $D(\mathrm{EQ}_n^2) = \Theta(n)$, we have $D(\mathrm{EQ}_n^k) = \Omega(n)$. Combining the bounds, $D(\mathrm{EQ}_n^k) = \Theta(n)$.

Next, we consider $\mathrm{NE}_n^k(x_1, \ldots, x_k)$. For the upper bound, consider a protocol where every player broadcasts their entire input x_i to a common blackboard or referee. The total communication is $k \cdot n$ bits. The referee then verifies if all strings are distinct. Thus, $D(\mathrm{NE}_n^k) = O(kn)$. For the lower bound, we reduce the problem of determining the logical AND of $m = \lfloor k/2 \rfloor$ independent instances of two-party nonequality to NE_n^k. Let Alice possess $U = (u_1, \ldots, u_m)$ and Bob possess $V = (v_1, \ldots, v_m)$, where $u_j, v_j \in \{0,1\}^n$. They wish to compute $F(U, V) = \bigwedge_{j=1}^m [u_j \neq v_j]$. It is well-established that the deterministic communication complexity of this problem is $\Omega(mn)$. We map this problem to the k-party number-in-hand model as follows: Alice simulates Players $1, \ldots, m$ and Bob simulates Players $m+1, \ldots, 2m$. If k is odd, the last player is simulated by Alice and given a fixed unique value. For each $j \in \{1, \ldots, m\}$, Alice sets the input for Player j as $x_j = \langle j, u_j \rangle$ (the index j concatenated with string u_j). Bob sets the input for Player $m+j$ as $x_{m+j} = \langle j, v_j \rangle$. By construction, x_i and x_l are distinct for any $i \neq l$ unless the indices match modulo m (i.e., we compare a player from Alice's set and the corresponding player from Bob's set). Specifically, $x_j = x_{m+j}$ if and only if $u_j = v_j$. Therefore, $\mathrm{NE}_n^k(x_1, \ldots, x_k) = 1$ if and only if $u_j \neq v_j$ for all $j \in \{1, \ldots, m\}$. Since any protocol for NE_n^k solves the m-fold nonequality problem, we have $D(\mathrm{NE}_n^k) \geq \Omega(mn) = \Omega(kn)$. Combining the bounds, $D(\mathrm{NE}_n^k) = \Theta(kn)$. □

H.1.5 Grok

H.2 Problem 2

H.2.1 Problem Statement

Theorem H.2. *For most Boolean functions $f : (\{0,1\}^n)^k \to \{0,1\}$, we have $D(f) = \Omega(n)$, where $D(f)$ denotes the deterministic communication complexity of f.*

Hint: Count the number of protocols of a given cost.

H.2.2 GPT

Proof. We begin by considering a Boolean function $f : (\{0,1\}^n)^k \to \{0,1\}$ and aim to show that its deterministic communication complexity $D(f) = \Omega(n)$ for most Boolean functions. To do this, we will count the number of protocols of a given communication cost and demonstrate that for most functions, any protocol must have a communication cost that grows with n.

Let the input of each party in the communication protocol be denoted as $x_1, x_2, \ldots, x_k \in \{0,1\}^n$. A communication protocol for f consists of a sequence of messages exchanged between two parties, say Alice and Bob, where Alice holds $x_1, x_2, \ldots, x_{k/2}$ and Bob holds $x_{k/2+1}, \ldots, x_k$, and the goal is to determine $f(x_1, x_2, \ldots, x_k)$.

Let C be the communication cost of a protocol, which is the total number of bits exchanged during the protocol. Suppose for the moment that there exists a protocol with cost $C = O(n)$, i.e., the number of bits exchanged is at most some constant multiple of n.

Next, we count the number of possible communication protocols of cost at most $O(n)$. There are at most $2^{O(n)}$ possible messages exchanged in such a protocol, and each message is a bit string of length at most $O(n)$. Hence, the total number of possible communication protocols with cost $O(n)$ is bounded by $2^{O(n)}$.

However, the function f can take on 2^{2^n} distinct values for $k = 2^n$ distinct inputs. Therefore, the number of possible distinct functions f is $2^{2^{2n}}$. In order for a protocol to correctly compute the function f, the communication protocol must be able to distinguish between all 2^{2^n} possible inputs.

Now, if there exists a protocol of cost $O(n)$ for most functions f, then the number of such protocols must be at least $2^{2^{2n}}$. But from the previous step, we have at most $2^{O(n)}$ possible protocols of cost $O(n)$, which gives a contradiction because $2^{O(n)}$ grows much slower than $2^{2^{2n}}$. Thus, for most Boolean functions f, there can be no protocol of communication cost $O(n)$.

Therefore, for most Boolean functions f, the deterministic communication complexity $D(f)$ must satisfy $D(f) = \Omega(n)$. □

H.2.3 Claude

Proof. We prove this result by counting the number of possible communication protocols with cost at most c and comparing it to the total number of Boolean functions on $(\{0,1\}^n)^k$.

First, we count the number of deterministic communication protocols with communication cost at most c. A deterministic protocol can be represented as a binary tree where each internal node is labeled with either a query by the first player or the second player, and each leaf is labeled with an output value in $\{0,1\}$.

For a protocol with cost at most c, the tree has depth at most c, and therefore has at most $2^{c+1} - 1$ nodes total and at most 2^c leaves. At each internal node, a player makes a query based on their input. The first player has an input in $\{0,1\}^n$ (assuming $k = 2$ for simplicity; the general case follows similarly), so there are at most 2^n possible queries. Similarly for the second player. Thus, each internal node can be labeled in at most 2^n ways (choosing which coordinate or function of the input to query). Each leaf can be labeled in 2 ways (output 0 or 1).

The total number of protocols with cost at most c is therefore at most

$$(2^n)^{2^{c+1}} \cdot 2^{2^c} \leq 2^{n \cdot 2^{c+1} + 2^c} = 2^{2^c(2n+1)}.$$

Now we count the total number of Boolean functions $f : (\{0,1\}^n)^2 \to \{0,1\}$. The domain has size $2^n \cdot 2^n = 2^{2n}$, so there are

$$2^{2^{2n}}$$

such functions.

For most functions to have $D(f) \geq c$, we need the number of functions computable with cost less than c to be a vanishing fraction of all functions. This requires

$$2^{2^c(2n+1)} \ll 2^{2^{2n}}.$$

Taking logarithms, we need

$$2^c(2n+1) \ll 2^{2n}.$$

If $c = \alpha n$ for some constant $\alpha < 1$, then

$$2^c(2n+1) = 2^{\alpha n}(2n+1) \ll 2^{2n}$$

for sufficiently large n, since $2^{\alpha n}$ grows much slower than 2^{2n} when $\alpha < 1$.

More precisely, if $c < n - \log n - \omega(1)$, then $2^c < 2^n/n$, and thus

$$2^c(2n+1) < \frac{2^n}{n}(2n+1) < 3 \cdot 2^n \ll 2^{2n}.$$

Therefore, the fraction of functions with $D(f) < n - \log n - \omega(1)$ is at most

$$\frac{2^{2^c(2n+1)}}{2^{2^{2n}}} = 2^{2^c(2n+1)-2^{2n}} \to 0$$

as $n \to \infty$.

This shows that for a random Boolean function $f : (\{0,1\}^n)^2 \to \{0,1\}$, with probability approaching 1 as $n \to \infty$, we have $D(f) = \Omega(n)$. □

H.2.4 Gemini

<think>**Defining the Proof's Scope**

I'm starting by carefully breaking down the request. My initial goal is to fully understand the theorem statement. I'm focusing on defining the scope and identifying the essential components involved, especially the deterministic communication complexity aspect of Boolean functions. Next, I need to understand all the constraints.

Understanding the Problem's Nuances

I've clarified the input and the assumed context. Now, I'm leaning towards the two-party model ($k = 2$) for communication complexity, given the lack of specific details. The input's notation confused me at first, with the k inputs of size n, but it makes more sense now. I'm focusing on why a standard counting argument should work here.

Developing the Argument's Logic

I'm starting to build a formal argument. Specifically, I'm defining $\mathcal{F}$ as the set of all Boolean functions and calculating its size. After that, I'm working to bound the number of communication protocols with a certain cost. I need to establish a clear connection between the number of functions and the number of protocols to get my desired bound.

Constructing the Communication Tree

I'm now sketching the core of the proof. My focus is on formally defining a deterministic communication protocol as a binary tree. I'm carefully analyzing how to bound the number of possible protocols of a given cost 'c', and then relating that bound to the total number of Boolean functions, leveraging the hint to guide my approach. The $k = 2$ case still feels like a good starting point.

Quantifying Protocol Possibilities

I'm now calculating the number of possible communication protocols of a given cost. I'm focusing on bounding the total number of trees, and also the functions chosen at the nodes. Using the two-party case as a reference, I am assessing how the number of protocols increases with the communication cost, aiming for the desired lower bound on the deterministic communication complexity, following the hint given.

Finalizing Protocol Bounds

I'm making a final refinement. I will formalize the bounds on the number of possible protocols based on the number of nodes, the choice of the speaking player at each node, and also the function that that player can use. I have to make sure to link all this to the total number of possible functions, so I can arrive at the desired result. Now, I should be able to write the conclusion.

Revisiting Node Labeling

I'm revisiting the node labeling for the communication tree to refine the counting. Now, I'm focusing on the number of possible functions each player can use at a node, which is a function of their private input. I need to make sure I accurately account for the player choice and the function selection to arrive at a solid protocol bound.

Formalizing Protocol Choices

I'm now carefully examining the internal node choices in the communication tree. For each node, I will factor the number of possible players, then the number of possible functions each of those players could employ, and finally the length of the binary string of the protocol. I'm focusing on rigorously bounding the protocol space size.

Refining Protocol Structure

I'm now carefully examining the internal node choices in the communication tree. For each node, I am considering the number of possible players, then the number of possible functions each player could

employ. This should allow me to accurately determine the protocol space size and link it to the total number of functions in order to prove the theorem.

Calculating Function Space Size

I'm now carefully calculating the size of the function space $\mathcal{F}$, which consists of all possible Boolean functions. Since the domain has size 2^{nk} and the codomain is $\{0,1\}$, the number of functions is $2^{2^{nk}}$. This establishes a crucial benchmark for the proof.

Revising Proof Structure

I'm now revising the proof structure based on the previous calculations, focusing on clear explanations. I'm carefully detailing each component of a protocol: tree depth, player choice, function evaluation, and the final output. The goal is to provide a complete, self-contained argument.

Revising Proof Presentation

I'm now carefully structuring the proof. I'm starting by defining the function space and setting up the communication model based on the binary tree representation. Then, I'll bound the number of possible protocols of a given cost. The ultimate goal is to apply the counting argument to show most functions require high communication complexity.

Synthesizing the Core Argument

I'm now refining the key steps for the proof, aiming to provide a clear and concise argument to the expert audience. I'm focusing on defining the function and protocol spaces, and then showing that the number of protocols of cost c grows slower than the number of all possible functions when c is less than a certain threshold.

Formalizing the Argument's Logic

I'm now focusing on formalizing the proof's core components. I've begun by defining the function space and setting up the communication protocol model. Next, I plan to rigorously bound the number of communication protocols with a certain cost and then relate the bound to the total number of Boolean functions to conclude.

</think>

Proof. Let $\mathcal{F}$ be the set of all Boolean functions $f : (\{0,1\}^n)^k \to \{0,1\}$. The size of the domain is 2^{nk}, and thus the total number of such functions is

$$|\mathcal{F}| = 2^{2^{nk}}.$$

Let $\mathcal{P}_c$ denote the set of all deterministic communication protocols of cost at most c. A deterministic protocol of cost c can be represented by a binary tree of depth at most c. In this tree: 1. Each internal node is associated with one of the k players and a Boolean function on that player's n-bit input (determining the bit sent). 2. Each leaf is labeled with an output value in $\{0,1\}$.

A binary tree of depth c has at most 2^c leaves and at most $2^c - 1$ internal nodes. We bound the number of distinct protocols by counting the number of ways to label such a tree. For each internal node, there are k choices for the player, and 2^{2^n} choices for the function of that player's input. Thus, there are $k \cdot 2^{2^n}$ choices per internal node. For each leaf, there are 2 choices for the output value. Therefore, the number of protocols of cost c is bounded by:

$$|\mathcal{P}_c| \leq \left(k \cdot 2^{2^n}\right)^{2^c} \cdot 2^{2^c} = \left(2k \cdot 2^{2^n}\right)^{2^c}.$$

Taking the logarithm base 2, we have:

$$\log_2 |\mathcal{P}_c| \leq 2^c(\log_2(2k) + 2^n).$$

For sufficiently large n, $2^n > \log_2(2k)$, so we may simplify the bound to:

$$\log_2 |\mathcal{P}_c| \leq 2^c \cdot 2^{n+1} = 2^{c+n+1}.$$

We compare this to the number of functions $|\mathcal{F}|$. A function f has complexity $D(f) \leq c$ only if f can be computed by one of the protocols in $\mathcal{P}_c$. We consider the case where $c = n - 3$ (assuming $k \geq 2$). Substituting c into the inequality:

$$\log_2 |\mathcal{P}_{n-3}| \leq 2^{(n-3)+n+1} = 2^{2n-2}.$$

Since $k \geq 2$, the log of the total number of functions is $\log_2 |\mathcal{F}| = 2^{nk} \geq 2^{2n}$. Comparing the exponents:

$$\frac{|\mathcal{P}_{n-3}|}{|\mathcal{F}|} \leq \frac{2^{2^{2n-2}}}{2^{2^{2n}}} = 2^{2^{2n}/4 - 2^{2n}} = 2^{-\frac{3}{4}2^{2n}}.$$

This ratio approaches 0 as $n \to \infty$. Thus, the fraction of functions with deterministic communication complexity $D(f) \leq n-3$ is negligible. It follows that for most Boolean functions, $D(f) > n-3$, which implies $D(f) = \Omega(n)$. □

H.2.5 Grok

Proof. The total number of Boolean functions $f : (\{0,1\}^n)^k \to \{0,1\}$ is $2^{2^{kn}}$.

We show that the number of such functions with deterministic communication complexity $D(f) \leq c$ is at most $2^{2^{n+c+O(1)}}$, for a suitable choice of the $O(1)$ term depending on k.

This implies that when $c = o(n)$ (or more precisely, when $c \leq (k-2)n$ for fixed $k \geq 2$ and sufficiently large n), the number of low-complexity functions is negligible compared to $2^{2^{kn}}$, so $D(f) = \Omega(n)$ for almost all f.

To bound the number of computable functions, we upper bound the number of deterministic protocols with worst-case communication at most c bits in the blackboard model, where players write single bits sequentially.

The possible transcripts are binary strings of length at most c, and there are fewer than $m = 2^{c+1}$ such strings (including all prefixes).

A protocol is specified by:

- For each of the fewer than m prefixes of length less than c, a choice of which player in $[k]$ writes the next bit or to halt instead: at most $(k+1)^m$ ways.

- The communication tree has fewer than m nodes where a bit is written. For each such node (assigned to some player by the previous choice), the bit written depends on that player's input in $\{0,1\}^n$: 2^{2^n} choices per node, hence at most $(2^{2^n})^m$ ways overall.

- For each of the fewer than m possible halting transcripts, the output in $\{0,1\}$: at most 2^m ways.

The total number of protocols is thus at most

$$(k+1)^m \cdot (2^{2^n})^m \cdot 2^m = 2^{m(\log_2(k+1)+2^n+1)}.$$

With $m = 2^{c+1}$,

$$2^{m(\log_2(k+1)+2^n+1)} = 2^{2^{c+1}(2^n+\log_2(k+1)+1)} \leq 2^{2^{n+c+O_k(1)}},$$

where the $O_k(1)$ absorbs $\log_2(k+1)+1 \leq O_k(1)$.

Each protocol computes at most one function, so the number of functions with $D(f) \leq c$ is at most $2^{2^{n+c+O_k(1)}}$.

Now fix $k \geq 2$ and choose $c = \lfloor n/2 \rfloor$. Then $n + c + O_k(1) \leq n + n/2 + O_k(1) = (3/2)n + O_k(1)$.

For sufficiently large n,

$$2^{n+c+O_k(1)} \leq 2^{(3/2)n+O_k(1)} < 2^{kn},$$

since $kn \geq 2n > (3/2)n + O_k(1)$. Thus,

$$2^{2^{n+c+O_k(1)}} < 2^{2^{kn}},$$

and in fact vastly smaller. The fraction of functions with $D(f) \leq n/2$ is at most $2^{2^{n+c+O_k(1)}-2^{kn}} \ll 1$.

Therefore, $D(f) > n/2 = \Omega(n)$ for almost all Boolean functions f. □

H.3 Problem 3

H.3.1 Problem Statement

Theorem H.3 (Exercise 6.12)**.** *As in the two-party case, we can define the nondeterministic communication complexity of a function as the number of bits that the players need to exchange in order to be convinced that $f(x_1, \ldots, x_k) = 1$. Similarly, the co-nondeterministic communication complexity of a function is the number of bits that the players need to exchange in order to be convinced that $f(x_1, \ldots, x_k) = 0$. Prove that the nondeterministic communication complexity of E_M^k, for every fixed k, is $\theta(\log \xi_k(N))$, whereas the co-nondeterministic complexity is $\theta(\log \log \xi_k(N))$.*

H.3.2 GPT

Proof. Let $f(x_1, \ldots, x_k) = E_M^k(x_1, \ldots, x_k)$, where E_M^k is the k-fold repetition of the disjointness function. The problem asks to prove that the nondeterministic communication complexity of E_M^k is $\theta(\log \xi_k(N))$, and the co-nondeterministic communication complexity is $\theta(\log \log \xi_k(N))$, where $\xi_k(N)$ denotes the number of possible nondeterministic communication protocols for k-fold disjointness with N inputs.

Nondeterministic Communication Complexity

The nondeterministic communication complexity of a function f is the number of bits exchanged between the players in the worst-case scenario, assuming the existence of a nondeterministic protocol for verifying that $f(x_1, \ldots, x_k) = 1$. For the function E_M^k, the nondeterministic protocol verifies that the disjointness condition holds in the k-fold repetition of the function.

Each player can send a message that encodes a certificate of the solution. Specifically, for E_M^k, the protocol is based on nondeterministically guessing the positions where the input vectors disagree. Each guess can be verified by the other player with a logarithmic number of bits, corresponding to the positions of the disagreements.

Let $\xi_k(N)$ denote the maximum number of nondeterministic communication protocols for k-fold disjointness with N inputs. Since the complexity is logarithmic in the number of possible nondeterministic protocols, the nondeterministic communication complexity is given by:

$$\text{Nondet}(E_M^k) = \Theta(\log \xi_k(N))$$

Thus, we have proven that the nondeterministic communication complexity of E_M^k is $\theta(\log \xi_k(N))$.

Co-Nondeterministic Communication Complexity

The co-nondeterministic communication complexity of a function f is defined as the number of bits exchanged between the players in the worst-case scenario, assuming the existence of a nondeterministic protocol for verifying that $f(x_1, \ldots, x_k) = 0$. For the function E_M^k, the co-nondeterministic protocol verifies that the disjointness condition does not hold in the k-fold repetition of the function.

In the co-nondeterministic case, the protocol involves sending certificates that the disjointness does not hold in the k repetitions. Each certificate can be verified with a logarithmic number of bits, similar to the nondeterministic case, but since the protocol is designed to verify a negative outcome, the complexity is lower. Specifically, the co-nondeterministic complexity is proportional to the logarithm of the logarithm of the number of possible nondeterministic protocols.

Thus, the co-nondeterministic communication complexity of E_M^k is given by:

$$\text{Co-Nondet}(E_M^k) = \Theta(\log \log \xi_k(N))$$

This completes the proof that the co-nondeterministic communication complexity of E_M^k is $\theta(\log \log \xi_k(N))$.

□

H.3.3 Claude

Proof. We prove both bounds for the nondeterministic and co-nondeterministic communication complexity of E_M^k.

Nondeterministic Communication Complexity:

First, we establish the upper bound. To convince the players that $E_M^k(x_1, \ldots, x_k) = 1$, it suffices for a prover to provide a witness consisting of a common element $m \in [N]$ such that $m \in x_i$ for all $i \in [k]$. Each player i can verify that $m \in x_i$ locally. The number of bits required to specify such an element is $\lceil \log N \rceil = O(\log N)$. Since $\xi_k(N)$ denotes the maximum size of a k-uniform intersecting family on $[N]$, and by known results $\xi_k(N) = \Theta(N^{k-1})$ for fixed k, we have $\log \xi_k(N) = \Theta((k-1) \log N) = \Theta(\log N)$ for fixed k. Thus, the nondeterministic complexity is $O(\log \xi_k(N))$.

For the lower bound, consider the fooling set argument. Any nondeterministic protocol must distinguish between all pairs of inputs $(x_1, \ldots, x_k)$ and $(x'_1, \ldots, x'_k)$ where $E_M^k(x_1, \ldots, x_k) = 1$ but $E_M^k(x'_1, \ldots, x'_k) = 0$. The number of distinct k-tuples with non-empty intersection is at least $\xi_k(N)$,

since any maximal intersecting family achieves this size. By the nondeterministic communication complexity lower bound via certificate complexity, we need at least $\log \xi_k(N)$ bits to specify which accepting input we have. Therefore, the nondeterministic complexity is $\Omega(\log \xi_k(N))$.

Combining both bounds, we obtain that the nondeterministic communication complexity is $\Theta(\log \xi_k(N))$.

Co-nondeterministic Communication Complexity:

For the upper bound, to convince the players that $E_M^k(x_1, \ldots, x_k) = 0$, the prover can provide a certificate of disjointness. By Hall's theorem and its generalizations, it suffices to exhibit a partition argument. Specifically, the prover can provide an index $i \in [k]$ and an element $m \in [N]$ such that m is the unique element preventing intersection (or demonstrate a minimal certificate of disjointness). The key observation is that for k-wise disjointness, we need only specify which of $O(\log k)$ structural conditions fails, and this requires $O(\log \log N)$ bits when k is fixed and we use an optimal encoding. More precisely, by providing a hitting set separator of logarithmic size in the universe, we need $O(\log \log \xi_k(N))$ bits.

For the lower bound, consider that the co-nondeterministic complexity is related to the logarithm of the minimum certificate size for disjointness. For k-wise intersection, the minimum certificate showing disjointness requires specifying structural information about the failure of intersection. By a counting argument on the number of minimal disjoint configurations, this requires at least $\Omega(\log \log \xi_k(N))$ bits. Specifically, the number of distinct minimal certificates for disjointness is super-polynomial in $\log N$ but sub-polynomial in N, giving a certificate complexity of $\Theta(\log \log \xi_k(N))$.

Therefore, the co-nondeterministic communication complexity is $\Theta(\log \log \xi_k(N))$. □

H.3.4 Gemini

Proof. Let E_M^k denote the generalized equality function on k inputs $x_1, \ldots, x_k$, where each x_i is drawn from a domain $\mathcal{D}$ of size $S = \xi_k(N)$. The function is defined as:

$$E_M^k(x_1, \ldots, x_k) = \begin{cases} 1 & \text{if } x_1 = x_2 = \cdots = x_k \\ 0 & \text{otherwise} \end{cases}$$

We analyze the nondeterministic communication complexity $N^1(E_M^k)$ and the co-nondeterministic communication complexity $N^0(E_M^k)$. Note that in the number-on-the-forehead (NOF) model, Player i sees all inputs except x_i.

1. Nondeterministic Communication Complexity ($N^1(E_M^k)$)

Upper Bound: To verify that $E_M^k(x_1, \ldots, x_k) = 1$, the players must establish that all inputs are identical. A valid protocol is as follows: Player 1 nondeterministically guesses the value $v \in \mathcal{D}$ such that $x_1 = v$ and broadcasts v to all other players. This requires $\lceil \log S \rceil$ bits. Then, every Player j (for $j = 2, \ldots, k$) verifies that their local input x_j (which they see on Player j's forehead? No, Player j holds x_j or sees others? In NOF, Player j sees $x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_k$. To verify equality $x_1 = \cdots = x_k$, Player 1 sees $x_2, \ldots, x_k$. Player 1 can verify $x_2 = \cdots = x_k$. To verify x_1 matches, Player 2 (who sees x_1) must verify $x_1 = x_3$ etc. Correct Protocol: Player 1 broadcasts x_2 (since $x_1 = x_2$ if equal). No, Player 1 does not know x_1. Standard NOF Protocol for Equality: Player 1 checks if $x_2 = x_3 = \cdots = x_k$. If not, Player 1 aborts. If equal, Player 1 broadcasts the value $v = x_2$. This costs $\lceil \log S \rceil$ bits. Player 2 sees x_1. Player 2 verifies $x_1 = v$. Since Player 1 has already verified $x_2 = \cdots = x_k = v$, if Player 2 confirms $x_1 = v$, then $x_1 = \cdots = x_k$. Thus, $N^1(E_M^k) \leq \lceil \log S \rceil + O(1) = O(\log \xi_k(N))$.

Lower Bound: The set of 1-inputs is the diagonal $\Delta = \{(v, v, \ldots, v) \mid v \in \mathcal{D}\}$. $|\Delta| = S$. In communication complexity, $N^1(f) \geq \log C^1(f)$, where $C^1(f)$ is the cover number (minimum number of monochromatic 1-rectangles needed to cover the 1-inputs). A combinatorial rectangle in the k-party NOF model is a set $R = A_1 \times \cdots \times A_k$ where membership of x in R depends on whether $x \in A_i$ for all i, but the definition of cylinder intersections is strictly more general. A cylinder intersection is a set $R = \bigcap_{i=1}^k R_i$, where the indicator function of R_i does not depend on the i-th coordinate x_i. Consider a valid 1-rectangle R for E_M^k. If $(u, \ldots, u) \in R$ and $(v, \ldots, v) \in R$ with $u \neq v$, then by the property of cylinder intersections, the "mixed" input $(u, v, \ldots, v)$ must also be in R. Specifically, let $x = (u, \ldots, u)$ and $y = (v, \ldots, v)$. Since $x \in R_1$ (which doesn't depend on first coord) and $y \in R_1$,

any input with the same suffix as y is in R_1. Thus $(u, v, \ldots, v) \in R_1$. Similarly for other coordinates. However, $E_M^k(u, v, \ldots, v) = 0$ because $u \neq v$. Thus, a monochromatic 1-rectangle cannot contain two distinct diagonal elements. Therefore, we need at least $|\Delta| = S$ rectangles to cover all 1-inputs. $N^1(E_M^k) \geq \log S = \Omega(\log \xi_k(N))$.

Combining bounds, $N^1(E_M^k) = \Theta(\log \xi_k(N))$.

2. Co-Nondeterministic Communication Complexity ($N^0(E_M^k)$)

Upper Bound: To verify that $E_M^k(x_1, \ldots, x_k) = 0$, the players must demonstrate that there exist indices i, j such that $x_i \neq x_j$. Let the elements of $\mathcal{D}$ be represented as binary strings of length $L = \lceil \log S \rceil$. If $x_i \neq x_j$, they must differ in at least one bit position $p \in \{1, \ldots, L\}$. Protocol: 1. Nondeterministically guess two player indices $i, j \in \{1, \ldots, k\}$ and a bit position $p \in \{1, \ldots, L\}$. This requires $2 \log k + \log L$ bits. 2. Player m (where $m \neq i$ and $m \neq j$) broadcasts the p-th bit of x_i and the p-th bit of x_j. Note that since $k \geq 3$, there is always a third player who sees both x_i and x_j. If $k = 2$, they simply exchange the bit. 3. Players check if the bits differ. The communication cost is dominated by specifying the bit position p. Cost $\approx \log \log S$. Thus, $N^0(E_M^k) = O(\log \log S) = O(\log \log \xi_k(N))$.

Lower Bound: We reduce the two-party inequality problem to $N^0(E_M^k)$. Consider the case $k = 2$ (or embedding 2-party into k-party). The players Alice and Bob hold $x, y \in \{0, 1\}^L$ where $L = \log S$. They want to show $x \neq y$. It is a standard result that the nondeterministic complexity of determining $x \neq y$ (which is N^1(NE) or N^0(EQ)) is $\Omega(\log L)$. Specifically, to distinguish $x = y$ from $x \neq y$, a nondeterministic proof of inequality requires specifying the index of the bit difference, which takes $\log L$ bits. A fooling set argument for the complement of equality on L-bit strings shows a complexity of $\Omega(\log L)$. Since $L = \log S$, we have $N^0(E_M^k) = \Omega(\log \log S) = \Omega(\log \log \xi_k(N))$.

Combining bounds, $N^0(E_M^k) = \Theta(\log \log \xi_k(N))$. □

H.3.5 Grok

Proof. We prove the bounds for the nondeterministic and co-nondeterministic communication complexities separately.

First, consider the nondeterministic communication complexity of E_M^k, which verifies $E_M^k = 1$.

For the upper bound, there exists a collection of $\xi_k(N)$ 1-monochromatic product sets that covers all inputs where $E_M^k = 1$. The prover sends the index $j \in [\xi_k(N)]$ of the product set containing the input (requiring $O(\log \xi_k(N))$ bits). Each player i accepts if and only if their input x_i belongs to the corresponding component of the j-th product set. If $E_M^k = 1$, there exists some j for which the input lies in the j-th product set, so all players accept. If $E_M^k = 0$, the input lies in no such product set, so for every j some player rejects. Thus the nondeterministic complexity is $O(\log \xi_k(N))$.

For the lower bound, there exists a set S of $\xi_k(N)$ inputs where $E_M^k = 1$ with the property that no single 1-monochromatic product set contains more than one input from S. Therefore, any covering of the 1-inputs by 1-monochromatic product sets requires at least $\xi_k(N)$ such sets to cover S, so the nondeterministic complexity is $\Omega(\log \xi_k(N))$.

Thus the nondeterministic communication complexity is $\Theta(\log \xi_k(N))$.

Next, consider the co-nondeterministic communication complexity of E_M^k, which verifies $E_M^k = 0$.

For the upper bound, the 0-inputs admit a binary decision tree structure of depth $O(\log \log \xi_k(N))$ with the property that each leaf corresponds to a distinct reason for $E_M^k = 0$ and there are sufficiently many leaves. The prover sends the $O(\log \log \xi_k(N))$ bits describing the path from the root to the leaf corresponding to the reason why the current input satisfies $E_M^k = 0$. Each bit on the path corresponds to a choice that narrows the possible set for each player's input. Each player accepts if and only if every choice on the path is consistent with their input x_i. If $E_M^k = 0$, there exists a leaf reason and corresponding path for which the input is consistent at every step, so all players accept. If $E_M^k = 1$, there is no such leaf reason, so for every path some choice is inconsistent for at least one player, who rejects. Thus the co-nondeterministic complexity is $O(\log \log \xi_k(N))$.

For the lower bound, suppose for contradiction that there is a protocol with cost $o(\log \log \xi_k(N))$. Then there are $o(\log \xi_k(N))$ possible proofs. However, by the structure of E_M^k and the definition of $\xi_k(N)$, distinguishing the different cases where $E_M^k = 0$ requires at least $\Omega(\log \log \xi_k(N))$ bits in the worst case. Thus the co-nondeterministic complexity is $\Omega(\log \log \xi_k(N))$.

Thus the co-nondeterministic communication complexity is $\Theta(\log\log\xi_k(N))$. □

H.4 Problem 4

H.4.1 Problem Statement

Theorem H.4. *Prove that the randomized communication complexity of E_N^k, for every fixed k, is $\theta(\log\log\varepsilon_k(N))$ in the private coin model and $\theta(1)$ in the public coin model.*

H.4.2 GPT

Proof. Let E_N^k be the communication problem with N inputs, each from a set of size N, and a parameter k determining the complexity of the problem. We are tasked with analyzing the randomized communication complexity of this problem in both the private coin and public coin models.

Private Coin Model

In the private coin model, the communication complexity is determined by the ability of the players to utilize random bits independently. In this setting, each player has access to a sequence of independent random bits that are not shared with the other player. Let $\varepsilon_k(N)$ denote the error probability of the randomized protocol for E_N^k.

The key observation here is that the randomized communication complexity is dominated by the entropy of the problem. In particular, the private coin model allows for the use of randomization to reduce the number of bits exchanged. The communication complexity can thus be bounded by the amount of information that needs to be conveyed to achieve a solution with error probability at most $\varepsilon_k(N)$.

Since the error probability is $\varepsilon_k(N)$, we have the relation between the number of bits communicated and the error probability in the private coin model:

$$C_{\text{private}}(E_N^k) = \theta(\log\log\varepsilon_k(N))$$

This result follows from the fact that, in the private coin model, each player can use randomization to reduce the number of bits communicated logarithmically in terms of the error probability, and logarithmically again in terms of the size of the problem, resulting in a logarithmic double logarithmic complexity.

Public Coin Model

In the public coin model, the players are allowed to share random bits. This is a more powerful model than the private coin model, as the randomness is no longer private and can be shared between the players, leading to a reduction in communication complexity.

In the public coin model, the randomized communication complexity is typically constant for many problems, as the ability to share random bits can significantly reduce the need for communication. This is the case for E_N^k, where the randomization allows for the reduction of the communication complexity to a constant number of bits, independent of the size of the problem or the error probability.

Thus, for the public coin model, the randomized communication complexity is:

$$C_{\text{public}}(E_N^k) = \theta(1)$$

Conclusion

Combining these results, we have the following:

- In the private coin model, the randomized communication complexity of E_N^k is $\theta(\log\log\varepsilon_k(N))$. - In the public coin model, the randomized communication complexity of E_N^k is $\theta(1)$.

This completes the proof. □

H.4.3 Claude

Proof. We prove the randomized communication complexity bounds for E_N^k in both the private and public coin models.

Part 1: Private Coin Model.

We first establish the lower bound. Consider the function E_N^k which tests equality on k-tuples of elements from $[N]$. By a reduction from the standard equality function, we observe that any private coin

protocol for E_N^k must distinguish between the case where all k coordinates match versus at least one coordinate differs.

Let $\varepsilon_k(N)$ denote the number of distinct inputs to E_N^k, which satisfies $\varepsilon_k(N) = N^k$. By the information-theoretic argument, Alice and Bob must exchange enough bits to identify which of the $\varepsilon_k(N)$ possible equal pairs they hold (in expectation).

Using Newman's theorem for private randomness, any private coin protocol can be converted to a public coin protocol with $O(\log n)$ additional communication, where n is the input size. However, for the equality function, we have a more refined analysis.

By the probabilistic method and Yao's minimax principle, the private coin complexity satisfies:

$$R_{\text{priv}}(E_N^k) \geq \Omega(\log\log \varepsilon_k(N))$$

For the upper bound, we employ a fingerprinting protocol. Alice computes a hash function $h : [N]^k \to \{0,1\}^\ell$ where $\ell = O(\log\log \varepsilon_k(N))$, chosen using private randomness. Alice sends $h(x)$ to Bob, who compares it with $h(y)$. By choosing the hash function from a universal family with appropriate parameters, the collision probability for $x \neq y$ is at most $1/3$.

Since $\varepsilon_k(N) = N^k$, we have $\log\log \varepsilon_k(N) = \log(k \log N)$. The fingerprint length $\ell = O(\log\log \varepsilon_k(N))$ suffices to achieve constant error probability. Thus:

$$R_{\text{priv}}(E_N^k) = O(\log\log \varepsilon_k(N))$$

Combining the bounds yields $R_{\text{priv}}(E_N^k) = \Theta(\log\log \varepsilon_k(N))$.

Part 2: Public Coin Model.

For the upper bound, we use the public coin equality protocol. Alice and Bob share a random hash function $h : [N]^k \to \{0,1\}$ chosen from a strongly universal family. Alice computes $h(x)$ and Bob computes $h(y)$. They each send a single bit indicating their hash value, using $O(1)$ communication.

By the properties of strongly universal hashing, if $x \neq y$, then $\Pr[h(x) = h(y)] \leq 1/2$. By repeating this protocol $O(1)$ times with independent hash functions, the error probability can be reduced to any constant $\delta < 1/3$. Therefore:

$$R_{\text{pub}}(E_N^k) = O(1)$$

For the lower bound, we use the fact that even with public randomness, Alice and Bob must exchange at least one bit of information to determine equality with probability bounded away from $1/2$. Specifically, by an information-theoretic argument, if no communication occurs, the best strategy achieves success probability at most $1/N^k + (1 - 1/N^k) \cdot 1/2 < 2/3$ for sufficiently large N.

More formally, by considering the mutual information between the inputs and the protocol transcript, we have that at least $\Omega(1)$ bits must be communicated. Since the upper bound is $O(1)$, we conclude:

$$R_{\text{pub}}(E_N^k) = \Theta(1)$$

This completes the proof for both models. □

H.4.4 Gemini

Proof. Let $S = \varepsilon_k(N)$ denote the size of the domain from which the inputs $x_1, \ldots, x_k$ are drawn. Let $L = \lceil \log S \rceil$ be the length of the input strings in bits. We assume the standard Number-in-Hand (NIH) model.

Public Coin Model *Upper Bound:* The players use a public source of randomness to select a hash function h from a universal family $\mathcal{H} = \{h : \{0,1\}^L \to \{0,1\}^c\}$ for a sufficiently large constant c. Player 1 computes $h(x_1)$ and broadcasts it to all other players. This requires $c = O(1)$ bits. Each Player i (for $i = 2, \ldots, k$) checks if $h(x_i) = h(x_1)$ and broadcasts a confirmation bit. The total communication is $O(k) = O(1)$ since k is fixed. If $x_1 = \cdots = x_k$, the protocol always accepts. If inputs differ, the collision probability is bounded by a constant, ensuring the error probability is small. *Lower Bound:* Since the output depends on the inputs, at least one bit of communication is required. Thus, the complexity is $\Omega(1)$. Combining these, the randomized public coin complexity is $\Theta(1)$.

Private Coin Model *Upper Bound:* Player 1 chooses a random prime p in the range $[L^2, 2L^2]$ uniformly at random. This prime can be described using $O(\log L)$ bits. Player 1 computes the fingerprint $F = x_1 \bmod p$ and broadcasts the pair (p, F). The size of F is $O(\log p) = O(\log L)$ bits. Each Player i then verifies if $x_i \equiv F \pmod p$. If all verifications pass, they accept. The total communication is $O(\log L)$. Substituting $L = \Theta(\log \varepsilon_k(N))$, the complexity is $O(\log \log \varepsilon_k(N))$.

Lower Bound: We reduce the two-party equality problem EQ_L to E_N^k. Let Alice and Bob possess inputs $x, y \in \{0,1\}^L$ respectively. They simulate a protocol for E_N^k by partitioning the k players. Alice simulates Player 1 and Players $3, \ldots, k$, assigning inputs $x_1 = x$ and $x_i = x$ for $i \geq 3$. Bob simulates Player 2, assigning $x_2 = y$. The condition $E_N^k(x_1, \ldots, x_k) = 1$ holds if and only if $x = y$. Communication between Alice's set of players and Bob's player corresponds to communication in the two-party protocol. It is a standard result that the randomized private coin communication complexity of EQ_L is $\Omega(\log L)$. Therefore, the complexity of E_N^k is $\Omega(\log L) = \Omega(\log \log \varepsilon_k(N))$.

Combining the bounds, the randomized private coin complexity is $\Theta(\log \log \varepsilon_k(N))$. □

H.4.5 Grok

Proof. We prove the bounds separately for the public coin and private coin models.

For the public coin model, we show that the randomized communication complexity is $O(1)$, with constant error probability $1/3$.

Using the shared public randomness, the players agree on a function h drawn from a pairwise-independent hash family mapping the input domain to $[Ck^2]$ for a sufficiently large constant C, ensuring that for any distinct inputs $a \neq b$, $\Pr[h(a) = h(b)] = 1/(Ck^2)$.

Each player $i = 1, \ldots, k$ in turn writes $h(x_i)$ on the blackboard, requiring $O(\log k)$ bits per player and $O(1)$ bits in total since k is fixed.

The output is 1 if all written values are equal and 0 otherwise.

If $E_N^k(x_1, \ldots, x_k) = 1$, all x_i are equal, so all $h(x_i)$ are equal.

If $E_N^k(x_1, \ldots, x_k) = 0$, there exist at least two distinct x_i, x_j; by pairwise independence and the union bound over $\binom{k}{2}$ pairs, the probability that all $h(x_i)$ are equal is at most $\binom{k}{2}/(Ck^2) < 1/3$.

For the $\Omega(1)$ lower bound, note that E_N^k takes both values 0 and 1 on some inputs, and a standard information-theoretic argument or fooling set consideration shows that constant communication is necessary even with public coins and bounded error.

Thus the public coin complexity is $\Theta(1)$.

For the private coin model, we show that the randomized communication complexity is $\Theta(\log \log \varepsilon_k(N))$.

The upper bound $O(\log \log \varepsilon_k(N))$ follows from a recursive protocol exploiting the combinatorial structure underlying $\varepsilon_k(N)$.

The parameter $\varepsilon_k(N)$ admits a recursive structure allowing reduction to instances of substantially smaller parameter with constant communication per reduction step.

Specifically, there exists a constant-communication reduction that, with high probability over private coins, reduces an instance with parameter N to one with parameter such that repeated application yields exponentially decreasing parameter values after $O(\log \log \varepsilon_k(N))$ successful reductions.

Each reduction step uses $O(1)$ communication, yielding the total $O(\log \log \varepsilon_k(N))$.

At the base case, constant communication suffices.

The lower bound $\Omega(\log \log \varepsilon_k(N))$ is proved by contradiction using round elimination and amplification on the recursive structure.

Suppose there is a private-coin protocol with cost $o(\log \log \varepsilon_k(N))$.

By repeated round elimination (fixing the first message to its most common value, losing a constant factor in success probability, amplifiable by repetition at $O(1)$ cost per amplification due to fixed k), we obtain a protocol for a reduced-parameter instance with even smaller cost.

After $O(\log \log \varepsilon_k(N))$ such eliminations, we obtain a constant-cost protocol for a base instance requiring $\omega(1)$ cost, a contradiction.

Thus the private coin complexity is $\Theta(\log \log \varepsilon_k(N))$. □

H.5 Problem 5

H.5.1 Problem Statement

Theorem H.5 (Exercise 6.15)**.** *Let f be a function. For every distribution μ,*

(1) $D^{\mu}_{\frac{1}{2}-\epsilon}(f) \geq \log(2\epsilon/\textit{Disc}_{\mu}(f))$*; and*

(2) $R_{\frac{1}{2}-\epsilon}(f) \geq \log(2\epsilon/\textit{Disc}_{\mu}(f))$.

(Obviously this implies $D(f) = \Omega(\log(1/\textit{Disc}_{\mu}(f)))$.)

H.5.2 GPT

Proof. We are tasked with proving the two inequalities stated in the theorem, namely:

$$D^{\mu}_{\frac{1}{2}-\epsilon}(f) \geq \log\left(\frac{2\epsilon}{\text{Disc}_{\mu}(f)}\right)$$

and

$$R_{\frac{1}{2}-\epsilon}(f) \geq \log\left(\frac{2\epsilon}{\text{Disc}_{\mu}(f)}\right).$$

We will prove each inequality separately.

Part (1): Proof of $D^{\mu}_{\frac{1}{2}-\epsilon}(f) \geq \log\left(\frac{2\epsilon}{\text{Disc}_{\mu}(f)}\right)$

Let μ be a distribution. By definition, the discrepancy $\text{Disc}_{\mu}(f)$ is given by:

$$\text{Disc}_{\mu}(f) = \frac{1}{2}\left|\Pr_{x\sim\mu}[f(x)=1] - \Pr_{x\sim\mu}[f(x)=0]\right|.$$

This represents the absolute difference in the probabilities of $f(x)$ being 1 versus 0 under the distribution μ. Now, the quantity $D^{\mu}_{\frac{1}{2}-\epsilon}(f)$ refers to the smallest c such that for all distributions ν close enough to μ in terms of the total variation distance, we have:

$$\left|\Pr_{x\sim\nu}[f(x)=1] - \Pr_{x\sim\nu}[f(x)=0]\right| \leq c.$$

The key observation here is that the discrepancy $\text{Disc}_{\mu}(f)$ bounds the total variation distance between the distributions of $f(x)$ under μ. By standard properties of total variation and the definition of discrepancy, we can deduce the following relationship:

$$D^{\mu}_{\frac{1}{2}-\epsilon}(f) \geq \log\left(\frac{2\epsilon}{\text{Disc}_{\mu}(f)}\right).$$

Part (2): Proof of $R_{\frac{1}{2}-\epsilon}(f) \geq \log\left(\frac{2\epsilon}{\text{Disc}_{\mu}(f)}\right)$

Next, we consider the quantity $R_{\frac{1}{2}-\epsilon}(f)$, which refers to the minimum value r such that for every distribution ν within a $\frac{1}{2}-\epsilon$ margin from μ, we have:

$$\left|\Pr_{x\sim\nu}[f(x)=1] - \Pr_{x\sim\nu}[f(x)=0]\right| \geq r.$$

By similar reasoning to the previous part, this quantity is related to the discrepancy $\text{Disc}_{\mu}(f)$ in a manner that leads to the desired inequality. Specifically, the function f can be forced to behave in a way that ensures a lower bound on $R_{\frac{1}{2}-\epsilon}(f)$ given the discrepancy $\text{Disc}_{\mu}(f)$. The derivation follows from the relationship between the discrepancy, the total variation distance, and the notion of a robust learner, leading to the bound:

$$R_{\frac{1}{2}-\epsilon}(f) \geq \log\left(\frac{2\epsilon}{\text{Disc}_{\mu}(f)}\right).$$

Conclusion

From Parts (1) and (2), we have established both inequalities. Moreover, since both bounds involve $\log\left(\frac{2\epsilon}{\text{Disc}_\mu(f)}\right)$, it follows that:

$$D(f) = \Omega\left(\log\left(\frac{1}{\text{Disc}_\mu(f)}\right)\right).$$

Thus, the theorem is proven. □

H.5.3 Claude

Proof. We prove both parts of the theorem.

Part (1): We show that $D^\mu_{\frac{1}{2}-\epsilon}(f) \geq \log(2\epsilon/\text{Disc}_\mu(f))$.

Let π be any deterministic protocol computing f with error at most $\frac{1}{2} - \epsilon$ under distribution μ. By definition of distributional complexity, we have $D^\mu_{\frac{1}{2}-\epsilon}(f) \leq |\pi|$, where $|\pi|$ denotes the depth of the protocol tree.

The protocol π partitions the input space into at most $2^{|\pi|}$ rectangles (leaves of the protocol tree). For each rectangle R in this partition, the protocol outputs a fixed value, say $b_R \in \{0, 1\}$.

The error of the protocol under μ is

$$\Pr_{(x,y)\sim\mu}[\pi(x, y) \neq f(x, y)] = \sum_R \mu(R) \cdot \Pr_{(x,y)\in R}[f(x, y) \neq b_R].$$

Since the protocol has error at most $\frac{1}{2} - \epsilon$, we have

$$\sum_R \mu(R) \cdot \Pr_{(x,y)\in R}[f(x, y) = b_R] \geq \frac{1}{2} + \epsilon.$$

For each rectangle R, let $\mu_0(R) = \mu(R \cap f^{-1}(0))$ and $\mu_1(R) = \mu(R \cap f^{-1}(1))$. The protocol chooses b_R to maximize correctness, so

$$\mu(R) \cdot \Pr_{(x,y)\in R}[f(x, y) = b_R] = \max\{\mu_0(R), \mu_1(R)\}.$$

Therefore,

$$\sum_R \max\{\mu_0(R), \mu_1(R)\} \geq \frac{1}{2} + \epsilon.$$

By definition of discrepancy, $\text{Disc}_\mu(f) = \max_R |\mu_0(R) - \mu_1(R)|$, where the maximum is over all rectangles. For the rectangles in our partition,

$$|\mu_0(R) - \mu_1(R)| \leq \text{Disc}_\mu(f).$$

Since $\mu_0(R) + \mu_1(R) = \mu(R)$ and $\max\{\mu_0(R), \mu_1(R)\} = \frac{\mu(R)+|\mu_0(R)-\mu_1(R)|}{2}$, we have

$$\sum_R \max\{\mu_0(R), \mu_1(R)\} \leq \sum_R \frac{\mu(R) + \text{Disc}_\mu(f)}{2} = \frac{1}{2} + \frac{2^{|\pi|} \cdot \text{Disc}_\mu(f)}{2}.$$

Combining with our earlier inequality:

$$\frac{1}{2} + \epsilon \leq \frac{1}{2} + \frac{2^{|\pi|} \cdot \text{Disc}_\mu(f)}{2}.$$

This gives $2\epsilon \leq 2^{|\pi|} \cdot \text{Disc}_\mu(f)$, hence $2^{|\pi|} \geq 2\epsilon/\text{Disc}_\mu(f)$, and therefore $|\pi| \geq \log(2\epsilon/\text{Disc}_\mu(f))$.

Since this holds for any protocol π with error at most $\frac{1}{2} - \epsilon$, we conclude $D^\mu_{\frac{1}{2}-\epsilon}(f) \geq \log(2\epsilon/\text{Disc}_\mu(f))$.

Part (2): We show that $R_{\frac{1}{2}-\epsilon}(f) \geq \log(2\epsilon/\text{Disc}_\mu(f))$.

By Yao's minimax principle, for any distribution μ over inputs,

$$R_{\frac{1}{2}-\epsilon}(f) \geq D^{\mu}_{\frac{1}{2}-\epsilon}(f).$$

Applying part (1), we obtain

$$R_{\frac{1}{2}-\epsilon}(f) \geq D^{\mu}_{\frac{1}{2}-\epsilon}(f) \geq \log(2\epsilon/\mathrm{Disc}_{\mu}(f)).$$

This completes the proof. □

H.5.4 Gemini

Proof. We first prove statement (1). Let $c = D^{\mu}_{\frac{1}{2}-\epsilon}(f)$. By definition, there exists a deterministic protocol Π with communication cost at most c such that the error probability under distribution μ satisfies

$$\Pr_{(x,y)\sim\mu}[\Pi(x,y) \neq f(x,y)] \leq \frac{1}{2} - \epsilon.$$

The protocol Π partitions the input space $X \times Y$ into a set of disjoint combinatorial rectangles $\{R_1, R_2, \ldots, R_L\}$, where $L \leq 2^c$ is the number of leaves in the protocol tree. For each rectangle R_i, the protocol outputs a fixed value $z_i \in \{0,1\}$. Thus, for any $(x,y) \in R_i$, $\Pi(x,y) = z_i$.

Consider the correlation between the protocol output and the function value, defined as the expectation of the parity:

$$\mathbb{E}_{(x,y)\sim\mu}\left[(-1)^{\Pi(x,y)\oplus f(x,y)}\right] = \sum_{(x,y)\in X\times Y} \mu(x,y)(-1)^{\Pi(x,y)\oplus f(x,y)}.$$

Let A be the event that $\Pi(x,y) = f(x,y)$. We have $\Pr[A] \geq 1 - (\frac{1}{2} - \epsilon) = \frac{1}{2} + \epsilon$. The expectation can be rewritten as:

$$\sum_{(x,y)} \mu(x,y)(-1)^{\Pi(x,y)\oplus f(x,y)} = \Pr[A] \cdot (1) + (1 - \Pr[A]) \cdot (-1) = 2\Pr[A] - 1.$$

Substituting the lower bound for $\Pr[A]$, we obtain:

$$2\Pr[A] - 1 \geq 2\left(\frac{1}{2} + \epsilon\right) - 1 = 2\epsilon.$$

Next, we decompose the sum over the disjoint rectangles R_i defined by the protocol:

$$\sum_{(x,y)} \mu(x,y)(-1)^{\Pi(x,y)\oplus f(x,y)} = \sum_{i=1}^{L} \sum_{(x,y)\in R_i} \mu(x,y)(-1)^{z_i \oplus f(x,y)}.$$

Using the property that $(-1)^{a\oplus b} = (-1)^a(-1)^b$, we can factor out the constant output z_i for each rectangle:

$$\sum_{i=1}^{L} (-1)^{z_i} \left(\sum_{(x,y)\in R_i} \mu(x,y)(-1)^{f(x,y)} \right).$$

By the definition of the discrepancy of f under μ, denoted $\mathrm{Disc}_{\mu}(f)$, we have for any combinatorial rectangle R:

$$\left| \sum_{(x,y)\in R} \mu(x,y)(-1)^{f(x,y)} \right| \leq \mathrm{Disc}_{\mu}(f).$$

Applying the triangle inequality to our sum:

$$\left|\sum_{i=1}^{L}(-1)^{z_i}\sum_{(x,y)\in R_i}\mu(x,y)(-1)^{f(x,y)}\right| \leq \sum_{i=1}^{L}|(-1)^{z_i}|\left|\sum_{(x,y)\in R_i}\mu(x,y)(-1)^{f(x,y)}\right|.$$

Since $|(-1)^{z_i}| = 1$, this is bounded by:

$$\sum_{i=1}^{L}\text{Disc}_\mu(f) = L\cdot\text{Disc}_\mu(f).$$

Combining the lower bound derived from the accuracy of the protocol and the upper bound derived from the discrepancy:

$$2\epsilon \leq L\cdot\text{Disc}_\mu(f).$$

Since the number of leaves L is at most 2^c, we have:

$$2\epsilon \leq 2^c\cdot\text{Disc}_\mu(f) \implies 2^c \geq \frac{2\epsilon}{\text{Disc}_\mu(f)}.$$

Taking the logarithm of both sides yields:

$$c \geq \log\left(\frac{2\epsilon}{\text{Disc}_\mu(f)}\right).$$

This concludes the proof of statement (1).

To prove statement (2), we recall the relationship between randomized communication complexity and distributional communication complexity. By Yao's Minimax Principle (or the standard definition of randomized complexity as the worst case over inputs of the best distribution over deterministic protocols, which implies the lower bound via the easy direction of the Minimax theorem), for any error parameter δ and any distribution μ:

$$R_\delta(f) \geq D^\mu_\delta(f).$$

Setting $\delta = \frac{1}{2}-\epsilon$, we have:

$$R_{\frac{1}{2}-\epsilon}(f) \geq D^\mu_{\frac{1}{2}-\epsilon}(f).$$

Using the result from statement (1), we immediately obtain:

$$R_{\frac{1}{2}-\epsilon}(f) \geq \log\left(\frac{2\epsilon}{\text{Disc}_\mu(f)}\right).$$

□

H.5.5 Grok

Proof. Let $f : \mathcal{X}\times\mathcal{Y}\to\{0,1\}$ be a function, and let μ be a probability distribution on $\mathcal{X}\times\mathcal{Y}$. A *combinatorial rectangle* is a set $R = A\times B$ with $A\subseteq\mathcal{X}$ and $B\subseteq\mathcal{Y}$. Define the *discrepancy* of f with respect to μ by

$$\text{Disc}_\mu(f) = \max_R\left|\int_R (2f(x,y)-1)\,d\mu(x,y)\right|,$$

where the maximum is over all combinatorial rectangles R.

We first prove (1). Suppose there exists a deterministic protocol Π of cost at most c such that

$$\Pr_{(x,y)\sim\mu}[\Pi(x,y) = f(x,y)] \geq \frac{1}{2}+\epsilon.$$

The protocol tree has at most 2^c leaves, and the inputs reaching each leaf l form a combinatorial rectangle R_l. The sets $\{R_l\}$ partition $\mathcal{X}\times\mathcal{Y}$ (up to a μ-null set). On each R_l, Π outputs a constant value $b_l\in\{0,1\}$.

Define $g(x,y) = b_l$ if $(x,y) \in R_l$. Then

$$\Pr[\Pi(x,y) = f(x,y)] = \frac{1 + \mathbb{E}_\mu[(2f(x,y)-1)(2g(x,y)-1)]}{2} \geq \frac{1}{2} + \epsilon,$$

so

$$\left|\mathbb{E}_\mu[(2f-1)g \cdot 2 - (2f-1)]\right| = \left|\mathbb{E}_\mu[(2f-1)(2g-1)]\right| \geq 2\epsilon.$$

(We may flip the sign of g globally if necessary to make the expectation positive.)

Now,

$$\mathbb{E}_\mu[(2f-1)(2g-1)] = \sum_l (2b_l - 1) \int_{R_l} (2f-1)\, d\mu.$$

For each l,

$$\left|\int_{R_l} (2f-1)\, d\mu\right| \leq \mathrm{Disc}_\mu(f).$$

Thus,

$$\left|\mathbb{E}_\mu[(2f-1)(2g-1)]\right| \leq \sum_l \left|\int_{R_l} (2f-1)\, d\mu\right| \leq 2^c \cdot \mathrm{Disc}_\mu(f).$$

Combining yields $2\epsilon \leq 2^c \cdot \mathrm{Disc}_\mu(f)$, so

$$2^c \geq \frac{2\epsilon}{\mathrm{Disc}_\mu(f)} \implies c \geq \log\left(\frac{2\epsilon}{\mathrm{Disc}_\mu(f)}\right).$$

Hence $D^\mu_{\frac{1}{2}-\epsilon}(f) \geq \log(2\epsilon/\mathrm{Disc}_\mu(f))$.

For (2), suppose there exists a (public-coin) randomized protocol $\mathcal{P}$ of cost at most c such that for every input (x,y),

$$\Pr[\mathcal{P}(x,y) = f(x,y)] \geq \frac{1}{2} + \epsilon,$$

where the probability is over the public randomness. Then $\mathcal{P}$ is a distribution over deterministic protocols $\{\Pi_r\}$ (by fixing the public randomness r), each of cost at most c.

Fix the distribution μ. For every (x,y),

$$\mathbb{E}_r[(2\Pi_r(x,y)-1)(2f(x,y)-1)] \geq 2\epsilon.$$

Integrating over μ,

$$\mathbb{E}_r\mathbb{E}_\mu[(2\Pi_r - 1)(2f-1)] \geq 2\epsilon.$$

Thus there exists a fixed r such that the deterministic protocol Π_r (of cost at most c) satisfies

$$\Pr_{(x,y)\sim\mu}[\Pi_r(x,y) = f(x,y)] \geq \frac{1}{2} + \epsilon.$$

By part (1),

$$c \geq D^\mu_{\frac{1}{2}-\epsilon}(f) \geq \log\left(\frac{2\epsilon}{\mathrm{Disc}_\mu(f)}\right).$$

Hence $R_{\frac{1}{2}-\epsilon}(f) \geq \log(2\epsilon/\mathrm{Disc}_\mu(f))$. □

H.6 Problem 6

H.6.1 Problem Statement

Theorem H.6. *Let E^k_N be the exactly-N function of Example 6.11. Then $D^I(E^k_N) = O(k \cdot \log^k_5(N))$.*

H.6.2 GPT

Proof. We are tasked with proving that $D^I(E_N^k) = O(k \cdot \log_5^k(N))$ for the exactly-N function E_N^k given in Example 6.11.

First, recall that $D^I(E_N^k)$ represents the measure of the complexity of E_N^k in the context of the I-measure, which, by the definition of E_N^k from Example 6.11, involves analyzing the number of operations required for E_N^k to compute the exact output for an input of size N.

Let us define the computational cost function $C(k, N)$, which gives the number of operations required by E_N^k to compute the exact output. The goal is to show that $C(k, N) = O(k \cdot \log_5^k(N))$ as $N \to \infty$.

From the definition of E_N^k, we know that the function involves recursive computations that depend on the value of k. The recursion depth is related to the logarithmic growth of N, and each recursive step incurs a cost that grows polynomially in k.

At each recursive step, the cost of computation can be bounded by $O(k \cdot \log_5(N))$ since the logarithmic factor arises from the division of the problem size at each level of recursion. Moreover, there are at most k recursive levels, as each level of recursion depends on the parameter k.

Thus, the total cost can be bounded as follows:

$$C(k, N) \leq k \cdot O(\log_5^k(N)).$$

Therefore, $D^I(E_N^k)$, which corresponds to the total computational complexity of the function E_N^k, satisfies:

$$D^I(E_N^k) = O(k \cdot \log_5^k(N)).$$

This completes the proof. □

H.6.3 Claude

Proof. We proceed by induction on k to establish the bound $D^I(E_N^k) = O(k \cdot \log_5^k(N))$.

Base case: For $k = 1$, the function E_N^1 is the exactly-N function on a single variable. By Example 6.11, we have $D^I(E_N^1) = O(\log N)$. Since $\log_5^1(N) = \log N$, the bound $D^I(E_N^1) = O(1 \cdot \log_5^1(N))$ holds.

Inductive step: Assume that for some $k \geq 1$, we have $D^I(E_N^{k-1}) = O((k-1) \cdot \log_5^{k-1}(N))$. We must show that $D^I(E_N^k) = O(k \cdot \log_5^k(N))$.

By the construction in Example 6.11, the function E_N^k can be expressed as a composition of E_N^{k-1} functions applied to subcubes. Specifically, E_N^k partitions the input into blocks and uses E_N^{k-1} to count occurrences in each block, then applies E_M^1 to verify that exactly N blocks satisfy the condition, where $M = O(\sqrt{N})$.

The decision tree complexity satisfies the recurrence:

$$D^I(E_N^k) \leq D^I(E_M^1) + \max_i D^I(E_{N_i}^{k-1})$$

where $M = O(\sqrt{N})$ and $N_i \leq N$ for all i.

By the base case, $D^I(E_M^1) = O(\log M) = O(\log \sqrt{N}) = O(\log N)$.

By the inductive hypothesis:

$$D^I(E_{N_i}^{k-1}) = O((k-1) \cdot \log_5^{k-1}(N))$$

Therefore:

$$D^I(E_N^k) = O(\log N) + O((k-1) \cdot \log_5^{k-1}(N))$$

We now verify that this yields $O(k \cdot \log_5^k(N))$. Recall that $\log_5^k(N)$ denotes the k-fold iterated logarithm. We have:

$$\log_5^k(N) = \log(\log_5^{k-1}(N))$$

Since $\log N \leq \log_5^{k-1}(N)$ for sufficiently large N and $k \geq 2$, and using the fact that $\log_5^{k-1}(N) = o(\log_5^k(N) \cdot \text{poly}(k))$ for the iterated logarithm, we obtain:

$$\log N + (k-1) \cdot \log_5^{k-1}(N) \leq C \cdot k \cdot \log_5^k(N)$$

for some constant $C > 0$ and sufficiently large N.

Thus:

$$D^I(E_N^k) = O(k \cdot \log_5^k(N))$$

By induction, the theorem holds for all $k \geq 1$. □

H.6.4 Gemini

Proof. We consider the Exactly-N function $E_N^k(x_1, \ldots, x_k)$, where $x_i \in \{0, \ldots, N\}$ and the function evaluates to 1 if and only if $\sum_{i=1}^k x_i = N$. We work in the Number-On-The-Forehead (NOF) model with k players. The stated bound involves $\log_5^k(N)$, which we interpret in the context of Chandra, Furst, and Lipton (CFL) protocols as the k-th iterated logarithm (or a related slowly growing function arising from k rounds of logarithmic reduction).

We describe a deterministic protocol that operates in rounds. The core idea is to reduce the problem of verifying the sum of inputs of magnitude M to verifying the sum of inputs of magnitude $O(\log M)$.

Base Step: Let the current inputs be $y_1, \ldots, y_k$ where each $y_i \in \{0, \ldots, M\}$. The players wish to check if $\sum_{i=1}^k y_i = T$ for some target T. The players agree on a small modulus m (typically a constant or small polynomial in k). Each Player i, seeing all y_j for $j \neq i$, computes $S_{-i} = \sum_{j \neq i} y_j$. Player i broadcasts the value $y_i \bmod m$. Since Player i does not know y_i, they cannot compute this directly. However, they can infer information. Actually, the standard CFL reduction proceeds as follows: The equation $\sum y_i = T$ can be written as $\sum y_i = T$. Let b be a base (e.g., $b = 2$). The players can verify the sum bit-by-bit or digit-by-digit. Alternatively, utilizing the properties of the sum, if $\sum y_i = T$, then $\sum (y_i \bmod m) \equiv T \pmod m$. However, a more efficient reduction uses the "carry" vector. Let c be the carry generated by the sum. The players can uniquely determine the carries if they know the sum modulo some small value.

Consider the following reduction step on inputs of size M: The players assume $\sum y_i = T$. They compute $y_i \bmod m$ for a sufficiently large constant m (e.g., $m > k$). To communicate $y_i \bmod m$ efficiently: Player k knows T and $y_1, \ldots, y_{k-1}$. If the equality holds, then $y_k = T - \sum_{j=1}^{k-1} y_j$. Player k can check consistency with their view. To reduce the magnitude: The players express the sum in terms of carries. $\sum y_i = T \implies \sum \lfloor y_i/m \rfloor + \text{carries} = \lfloor T/m \rfloor$. The new inputs become the values $\lfloor y_i/m \rfloor$ (roughly y_i/m) or related carry information. A more direct reduction due to Chandra, Furst, and Lipton allows reducing the bit-length of the inputs from L to $\log L$ in one round using $O(k)$ bits of communication. Let $L = \log M$. The players treat their inputs as integers. They verify the sum modulo primes $p_1, \ldots, p_t$ such that their product exceeds kM (not efficient) or using the hashing property. The specific CFL protocol for Exactly-N yields a recurrence for the communication complexity $D(L)$ on L-bit integers:

$$D(L) \leq D(\log L) + O(k)$$

This recurrence holds because to verify the sum of L-bit numbers, it suffices to verify the sum of their checksums (which are $\log L$ bits) and the consistency of the carries. Applying this reduction k times (or recursively), the input size L reduces as $L \to \log L \to \log \log L \ldots$. Let $\log^{(i)} N$ denote the i-th iterated logarithm. After k rounds (or a number of rounds related to $\log^* N$), the input size becomes constant. The total communication complexity is the sum of the communication in each round.

$$D(E_N^k) = \sum_{r=1}^{k} O(k) + O(1) \approx O(k^2) \quad \text{if reducing to constant in k steps?}$$

Actually, the standard result states that with k players, one can handle inputs up to size described by the $(k-1)$-th tower of exponentials with constant cost, or conversely, for input N, the cost involves the inverse of this tower function, often denoted $\lambda_k(N)$ or $\log^{(k)} N$. Specifically, the complexity is $O(k \cdot \log^{(k-1)} N)$. Matching the notation in the statement: $O(k \cdot \log_5^k(N))$. Assuming $\log_5^k(N)$ represents the k-times iterated logarithm (base 5), the derivation follows the sequence of $O(k)$ cost reductions, each applying a logarithm to the input size.

Therefore, $D^I(E_N^k) = O(k \cdot \log_5^k(N))$. □

H.6.5 Grok

Proof. We prove the upper bound by induction on k. There exists a universal constant $c > 0$ such that the deterministic communication complexity $D^I(E_N^k)$ is at most $ck(\log_5 N)^k$ for all $k \geq 1$ and all $N \geq 2$.

For the base cases $k \leq 10$ (say), we use the following explicit protocol: each player i computes $d_i = N - \sum_{j \neq i} x_j$ (treating invalid values, e.g., negative or exceeding the domain, as a special symbol). Each player sends d_i to the blackboard, requiring $O(\log N)$ bits per player. All players then locally compute the received values $\{d_1, \ldots, d_k\}$ and check if their sum equals N. As shown in the analysis, this correctly computes E_N^k with total communication $O(k \log N)$, which is $O(k(\log_5 N)^k)$ for k bounded (with c chosen sufficiently large). Assume now $k > 10$ and the statement holds for all smaller numbers of players (with the same c).

The players proceed as follows. They fix player k in advance. Players $1, \ldots, k-1$ all see x_k and thus compute $N' = N - x_k$. Note that $0 \leq N' \leq N$ (adjusting the domain conventions if necessary to ensure the promise holds for the reduced instance).

Players $1, \ldots, k-1$ then execute the induction hypothesis protocol for the $(k-1)$-player instance $E_{N'}^{k-1}$ on inputs $x_1, \ldots, x_{k-1}$ (in the NOF model restricted to these players). This correctly determines whether $\sum_{i=1}^{k-1} x_i = N'$, which is equivalent to $\sum_{i=1}^{k} x_i = N$. Player k sends nothing. The total communication is at most $c(k-1)(\log_5 N')^{k-1} \leq c(k-1)(\log_5 N)^{k-1}$.

We have

$$c(k-1)(\log_5 N)^{k-1} \leq ck(\log_5 N)^k$$

for sufficiently large c, since

$$(k-1)(\log_5 N)^{k-1} \leq k(\log_5 N)^k / (\log_5 N) = k(\log_5 N)^{k-1}$$

and $\log_5 N \geq 1$ for $N \geq 2$, with strict room for larger c to handle constants and small cases absorbed in the base.

Thus the induction holds, and $D^I(E_N^k) = O(k(\log_5 N)^k)$. □

H.7 Problem 7

H.7.1 Problem Statement

Problem H.7 (6.21)**.** *For some explicit function $f\colon (\{0,1\}^n)^k \to \{0,1\}$, prove a super-logarithmic lower bound on $D^{\|}(f)$ with $k \geq \log n$ parties.*

H.7.2 GPT

Proof. We are tasked with proving a super-logarithmic lower bound on the decision tree complexity $D^{\|}(f)$ of a function $f\colon (\{0,1\}^n)^k \to \{0,1\}$, where $k \geq \log n$. The decision tree complexity $D^{\|}(f)$ is the minimum number of queries to the input bits required to compute f, where each query can be to any bit of the input, and we are allowed to query the bits of the input vector in parallel.

Let $x_1, x_2, \ldots, x_k \in \{0,1\}^n$ be the input vectors, so the function f is defined on the k-tuple $(x_1, x_2, \ldots, x_k)$. Since $k \geq \log n$, the number of different input sets $(x_1, x_2, \ldots, x_k)$ is at least $2^{k \cdot n}$. We aim to show that any decision tree for f must have a super-logarithmic depth in terms of n and k.

Consider a decision tree for f. Each internal node of the tree corresponds to a query on some bit of the input, and the leaves correspond to possible outputs of the function. The total number of leaves of the tree corresponds to the number of distinct possible outputs of f, and since f maps to $\{0,1\}$, there are at most 2 possible outputs at each leaf.

Since f is defined on k input vectors of length n, the number of possible inputs to the function is $2^{k \cdot n}$. Each leaf of the decision tree corresponds to one of these possible inputs, and therefore, the number of leaves in the decision tree must be at least $2^{k \cdot n}$.

A decision tree with L leaves must have at least $\log L$ depth. Since the number of leaves is at least $2^{k \cdot n}$, the depth of the decision tree must be at least $\log(2^{k \cdot n}) = k \cdot n$. Hence, the decision tree complexity of f must satisfy

$$D^{\|}(f) \geq k \cdot n.$$

This establishes a lower bound on the decision tree complexity of f in terms of k and n. Since $k \geq \log n$, we have

$$D^{\parallel}(f) \geq k \cdot n \geq \log n \cdot n.$$

Thus, the decision tree complexity $D^{\parallel}(f)$ is super-logarithmic in n when $k \geq \log n$.

This completes the proof. □

H.7.3 Claude

Proof. We prove a super-logarithmic lower bound on the number-on-forehead (NOF) communication complexity for an explicit function with $k = \log n$ parties.

Let $k = \log n$ and consider the function $f\colon (\{0,1\}^n)^k \to \{0,1\}$ defined by the generalized inner product (GIP) function:

$$\text{GIP}(x_1, \ldots, x_k) = \bigoplus_{i_1,\ldots,i_k \in [n]} \left(\bigwedge_{j=1}^{k} x_j[i_j] \right)$$

where $x_j[i_j]$ denotes the i_j-th bit of the j-th player's input, and $\bigoplus$ denotes XOR.

We establish a lower bound using the discrepancy method. For any protocol π computing f with communication cost c, there exists a rectangle R in the partition induced by π with $|R| \geq 2^{kn}/2^c$.

For the NOF model with k parties, we use the fact that each party sees all inputs except their own. The key observation is that the GIP function has exponentially small discrepancy with respect to the uniform distribution.

Define the discrepancy of f as:

$$\text{disc}(f) = \max_R \left| \Pr_{(x_1,\ldots,x_k)\sim U}[f(x_1, \ldots, x_k) = 1 \mid (x_1, \ldots, x_k) \in R] - \frac{1}{2} \right|$$

where the maximum is over all combinatorial rectangles R in the NOF model.

By a counting argument, for $k = \log n$ parties, the GIP function satisfies:

$$\text{disc}(f) \leq 2^{-\Omega(n/k)} = 2^{-\Omega(n/\log n)}$$

This follows from the fact that fixing all but one player's input leaves a linear function in the remaining player's input, and the number of such constraints grows exponentially.

By the discrepancy lower bound for NOF communication complexity, any protocol computing f must satisfy:

$$2^c \geq \frac{1}{2\text{disc}(f)} \geq 2^{\Omega(n/\log n)}$$

Therefore:

$$D^{\parallel}(f) = c \geq \Omega\left(\frac{n}{\log n}\right)$$

Since n can be arbitrarily large, we have $\frac{n}{\log n} = \omega(\log n)$, which establishes the super-logarithmic lower bound.

More precisely, for sufficiently large n, we have:

$$D^{\parallel}(f) \geq \frac{n}{C \log n}$$

for some constant $C > 0$, which is super-logarithmic in the input size $kn = n \log n$. □

H.7.4 Gemini

Proof. We prove the existence of a super-logarithmic lower bound for the Generalized Inner Product function (GIP) in the simultaneous number-in-hand (NIH) communication model.

Let $f : (\{0,1\}^n)^k \to \{0,1\}$ be the Generalized Inner Product function, defined as:

$$\mathrm{GIP}_n^k(x_1, \ldots, x_k) = \bigoplus_{j=1}^{n} \left(\bigwedge_{i=1}^{k} x_{i,j} \right)$$

where $x_{i,j}$ denotes the j-th bit of the string x_i.

We consider the simultaneous message passing model where each of the k players $P_1, \ldots, P_k$ holds input x_i and sends a single message m_i to a referee. The referee determines the output based on $m_1, \ldots, m_k$. We reduce the two-party Inner Product modulo 2 (IP_n) problem to GIP_n^k.

Recall that $\mathrm{IP}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is defined as $\mathrm{IP}_n(u, v) = \bigoplus_{j=1}^{n}(u_j \wedge v_j)$. Let Π be a simultaneous protocol for GIP_n^k with communication cost C. We construct a simultaneous protocol Π' for IP_n. Let Alice hold $u \in \{0,1\}^n$ and Bob hold $v \in \{0,1\}^n$. They simulate the players in Π as follows: Alice simulates player P_1 with input $x_1 = u$. Bob simulates player P_2 with input $x_2 = v$. For players $P_3, \ldots, P_k$, we fix their inputs to be the all-ones string $\vec{1} = (1, \ldots, 1) \in \{0,1\}^n$. Since these inputs are fixed and known, Alice and Bob can deterministically generate the messages $m_3, \ldots, m_k$ that players $P_3, \ldots, P_k$ would send (or these can be treated as constants by the referee). Alice computes m_1 based on x_1 and Bob computes m_2 based on x_2. They send m_1 and m_2 to the referee. The referee computes the output of Π using messages m_1, m_2 and the pre-computed/fixed messages $m_3, \ldots, m_k$.

The output of the simulated protocol is:

$$\mathrm{GIP}_n^k(u, v, \vec{1}, \ldots, \vec{1}) = \bigoplus_{j=1}^{n} \left(u_j \wedge v_j \wedge \bigwedge_{i=3}^{k} 1 \right) = \bigoplus_{j=1}^{n} (u_j \wedge v_j) = \mathrm{IP}_n(u, v).$$

Thus, Π' correctly computes IP_n. The communication cost of Π' is at most C. It is a standard result in communication complexity that the deterministic simultaneous communication complexity of the Inner Product function is linear in n. Specifically, $D^{\|}(\mathrm{IP}_n) = \Theta(n)$. Consequently, $D^{\|}(\mathrm{GIP}_n^k) \geq D^{\|}(\mathrm{IP}_n) = \Omega(n)$.

Since n is the parameter defining the input size of the strings, a lower bound of $\Omega(n)$ is super-logarithmic in n. This bound holds for any $k \geq 2$, including the case $k \geq \log n$.

□

H.7.5 Grok

Proof. Although proving super-logarithmic lower bounds for explicit functions in the number-on-the-forehead model with $k \geq \log n$ parties remains open, a standard counting argument shows that there exists a (non-explicit) function $f : (\{0,1\}^n)^k \to \{0,1\}$ with $D^{\|}(f) = \Omega(n)$.

The total number of possible functions f is $2^{2^{kn}}$.

We bound the number of functions computable by deterministic protocols with communication cost at most c.

The possible transcripts are binary strings of length at most c, so there are $m < 2^{c+1}$ possible transcripts (considering all prefixes as potential halting points).

A protocol is specified by:

- For each of the $O(m)$ internal nodes of the communication tree, choosing which player speaks next or to halt: at most $(k+1)^{O(m)}$ choices.

- For each speaking node (fewer than m), the bit sent as a function of that player's n-bit input: at most $(2^{2^n})^{O(m)}$ choices.

- For each of the $O(m)$ possible halting transcripts, the output bit: at most $2^{O(m)}$ choices.

The total number of protocols is at most

$$2^{O(m \log k + m2^n + m)} = 2^{O(2^c(2^n + \log k + 1))}.$$

Each protocol computes one function, so the number of functions with $D^{\parallel}(f) \le c$ is at most $2^{O(2^c 2^n)}$.
Set $c = n/4$. Then the number is $2^{O(2^{n/4}2^n)} = 2^{O(2^{5n/4})}$.
Since $k \ge \log n$, we have $kn \ge n \log n$, so $2^{kn} \ge 2^{n \log n}$.
For sufficiently large n,

$$O(2^{5n/4}) < 2^{n \log n} \le 2^{kn},$$

so

$$2^{O(2^{5n/4})} \ll 2^{2^{kn}}.$$

Thus the fraction of functions with $D^{\parallel}(f) \le n/4$ is negligible, and there exist functions (in fact, almost all) with $D^{\parallel}(f) = \Omega(n) = \omega(\log n)$.
This establishes a super-logarithmic lower bound for some (non-explicit) function f. □

H.8 Problem 8

H.8.1 Problem Statement

Theorem H.8 (Exercise 6.23)**.** $\mathcal{D}_{(INDEX)} = \theta(\log n)$. *That is, k parties without the restriction to simultaneous protocols can do better than the lower bound generalized from Exercise 4.21.*

H.8.2 GPT

Proof. Let $\mathcal{D}_{(\text{INDEX})}$ denote the communication complexity of the INDEX problem for k parties. We are tasked with proving that $\mathcal{D}_{(\text{INDEX})} = \theta(\log n)$, where n is the size of the input, and the bound is achieved without the restriction to simultaneous protocols, implying no simultaneous communication among the parties.

First, recall the result from Exercise 4.21, which provides a lower bound for the communication complexity of k-party protocols in the context of the INDEX problem. The lower bound from Exercise 4.21 is based on the fact that the number of possible inputs is n and that any protocol must distinguish between all n possible inputs. Since each message sent in the protocol conveys at most one bit of information, the total number of bits exchanged must be at least $\log n$ to distinguish between all possible inputs.

Thus, the lower bound from Exercise 4.21 gives the following inequality for any protocol:

$$\mathcal{D}_{(\text{INDEX})} \ge \log n.$$

Now, consider the scenario where k parties are involved in the protocol. Without the restriction to simultaneous protocols, each party can communicate in a sequential manner. This allows for a greater degree of freedom in how the information is exchanged between the parties. Specifically, each party can receive information from the others before responding, which is not allowed in simultaneous protocols.

Given this setup, we can construct a communication protocol that performs better than the lower bound established in Exercise 4.21. In particular, we can design a protocol in which the communication complexity is reduced to $\theta(\log n)$, which means that the total number of bits exchanged scales logarithmically with n.

To establish the upper bound, we consider the following: for each of the k parties, the protocol involves exchanging messages that allow the parties to progressively narrow down the possible inputs. Since each message reveals only partial information about the input, the number of bits required to uniquely identify the input across all parties is proportional to $\log n$. Therefore, the total communication complexity of the protocol is bounded above by a constant multiple of $\log n$.

Thus, combining the lower bound and the upper bound, we conclude that the communication complexity of the k-party INDEX problem is asymptotically $\theta(\log n)$. This completes the proof.

□

H.8.3 Claude

Proof. We prove that the k-party communication complexity of the INDEX function satisfies $\mathcal{D}_{(\text{INDEX})} = \Theta(\log n)$.

Upper Bound: We first show that $\mathcal{D}_{(\text{INDEX})} = O(\log n)$.

Consider the following protocol for k parties where party i holds a portion of the index $j \in [n]$ represented in binary. Specifically, we distribute the $\log n$ bits of j among the k parties, and party k holds the database string $x \in \{0,1\}^n$.

The parties proceed sequentially to reconstruct the index j. Party 1 sends its portion of the index bits to party 2, party 2 appends its bits and sends to party 3, and so forth. After $k-1$ rounds, party k has received all $\log n$ bits of the index j. Party k then computes x_j and announces the result.

The total communication is at most $(k-1)\cdot\log n$ bits, which is $O(\log n)$ when k is constant. Even when k varies, the communication remains $O(k \log n)$. For the standard setting where the index is distributed and $k = O(1)$, we have $\mathcal{D}_{(\text{INDEX})} = O(\log n)$.

Lower Bound: We now show that $\mathcal{D}_{(\text{INDEX})} = \Omega(\log n)$.

Consider any deterministic protocol Π that computes INDEX. The protocol must distinguish between all n possible values of the index j.

For each possible input configuration, the protocol produces a transcript. Since the output must correctly identify x_j for all possible databases x, and there are n possible indices, the protocol must be able to produce at least n distinct outcomes based on the communication.

By a counting argument, if the total communication is c bits, then there are at most 2^c possible transcripts. For the protocol to correctly compute INDEX for all inputs, we require that for each index $j \in [n]$, there exists a transcript that leads to the correct output.

More formally, consider the information-theoretic argument: party k (holding the database) must learn the complete index j to output x_j. The index j contains $\log n$ bits of information. By the information transmission argument, at least $\log n$ bits must be communicated to party k (or to some party that computes the output).

Therefore, $\mathcal{D}_{(\text{INDEX})} \geq \log n$, giving us $\mathcal{D}_{(\text{INDEX})} = \Omega(\log n)$.

Comparison with Simultaneous Protocols: The lower bound from Exercise 4.21 for simultaneous protocols gives $\mathcal{D}^{\text{sim}}_{(\text{INDEX})} = \Omega(n)$, since in the simultaneous model, each party must send information to a referee without coordination, requiring each party to potentially send enough information to cover all possible contributions from other parties.

In contrast, the non-simultaneous k-party protocol allows sequential communication, enabling parties to build upon previous messages. This coordination reduces the complexity from $\Omega(n)$ to $\Theta(\log n)$, demonstrating that k parties without the simultaneity restriction can indeed perform significantly better.

Combining the upper and lower bounds, we conclude that $\mathcal{D}_{(\text{INDEX})} = \Theta(\log n)$. □

H.8.4 Gemini

Proof. We consider the multiparty version of the Index function, denoted INDEX^k_n, where Player 1 holds a string $x \in \{0,1\}^n$ and Player 2 holds an index $i \in \{1,\ldots,n\}$. The other $k-2$ players have no inputs (or irrelevant inputs). The goal is to compute x_i.

Upper Bound: Consider the following deterministic protocol Π: Player 2 encodes the index i as a binary string of length $\lceil \log n \rceil$ and sends it to Player 1. Player 1 receives i, retrieves the bit x_i from their input x, and broadcasts the result (1 bit). The total communication cost is $\lceil \log n \rceil + 1$. Thus, $\mathcal{D}(\text{INDEX}) = O(\log n)$.

Lower Bound: We show that $\mathcal{D}(\text{INDEX}) = \Omega(\log n)$ by reducing to the two-party case. Any k-party protocol for INDEX^k_n implies a two-party protocol between Alice (holding x) and Bob (holding i). Suppose there exists a protocol with communication cost $C < \log n$. The number of distinct communication transcripts (sequences of messages exchanged) is at most $2^C < n$. By the Pigeonhole Principle, there exist two distinct indices $j, k \in \{1,\ldots,n\}$ with $j \neq k$ such that the protocol generates the exact same transcript T for the input pairs (x, j) and (x, k)? This reasoning is subtle because the transcript depends on x as well. Instead, consider the dependency on the index i. In any valid protocol where Player 1 eventually learns i (or sufficient information to select x_i), information about i must flow to

Player 1. Consider the projection of the protocol where Alice holds $x = 0^n$ (all zeros). Bob holds index i. The sequence of messages sent by Bob must distinguish every index $i \in \{1, \ldots, n\}$. If not, suppose indices $j \neq k$ produce the same sequence of messages from Bob to Alice (and Alice sends the same responses, since her input is fixed). Alice, having the same view for inputs j and k, must produce the same output bit. However, the protocol must work for *any* x. Let the transcript determined by Bob's index i and Alice's input x be $\tau(x, i)$. The final output is a function $out(\tau(x, i))$. If the communication from Bob to Alice is less than $\log n$ bits, Bob cannot distinguish all n indices to Alice. Specifically, consider a one-way protocol (general interaction can be bounded similarly via fooling sets or observing that Alice must distinguish n states of Bob to index the correct bit). More formally, let the protocol define a partition of the index space $\{1, \ldots, n\}$ based on the messages Bob sends. If the total communication is $C < \log n$, there exist distinct indices j, k that are indistinguishable to Alice (Bob sends the same messages for both, assuming Alice's messages are fixed or the conversation tree path is identical). Let x be a string such that $x_j = 0$ and $x_k = 1$. Since Alice cannot distinguish whether Bob holds j or k, she must output the same value b for both cases. However, correct correctness requires the output to be $x_j = 0$ in the first case and $x_k = 1$ in the second case. This is a contradiction. Thus, the communication complexity must be at least $\log n$.

Combining the bounds, $\mathcal{D}(\text{INDEX}) = \Theta(\log n)$. □

H.8.5 Grok

Proof. The index function INDEX_n on k parties (with $k \geq 2$) is defined as follows: party 1 receives input $x \in \{0,1\}^n$, party 2 receives input $i \in [n]$, and parties 3 to k receive no input (or a fixed dummy input). The goal is to compute $\text{INDEX}_n(x, i) = x_i$.

We first prove the upper bound $\mathcal{D}(\text{INDEX}_n) = O(\log n)$.

Party 2 writes the binary representation of i on the blackboard, using at most $\lceil \log_2 n \rceil$ bits. Party 1 reads this value, computes the bit x_i, and writes x_i on the blackboard using 1 bit. All parties now see x_i on the blackboard. The total communication is $O(\log n)$.

We next prove the lower bound $\mathcal{D}(\text{INDEX}_n) = \Omega(\log n)$. Fix the inputs of parties 3 to k to be dummy. The problem now reduces to computing the two-party index function between parties 1 and 2.

Consider the n inputs (x^e, e) for $e \in [n]$, where $x^e \in \{0,1\}^n$ is the vector with 1 in position e and 0 elsewhere (and dummy inputs for the remaining parties). On each such input, $\text{INDEX}_n(x^e, e) = 1$.

Suppose a combinatorial rectangle R contains two distinct such inputs (x^e, e) and $(x^{e'}, e')$ with $e \neq e'$. Then R also contains the crossed inputs (x^e, e') and $(x^{e'}, e)$. However,

$$\text{INDEX}_n(x^e, e') = (x^e)_{e'} = 0, \quad \text{INDEX}_n(x^{e'}, e) = (x^{e'})_e = 0.$$

Thus R contains inputs with function values 0 and 1, so R is not monochromatic. It follows that no monochromatic rectangle contains more than one of these n inputs, all of which have function value 1.

Any deterministic protocol partitions the input space into at most 2^c monochromatic rectangles, where c is the communication cost. To cover the n inputs of function value 1, at least n monochromatic 1-rectangles are required. Thus $2^c \geq n$, so $c \geq \log_2 n = \Omega(\log n)$.

Therefore, $\mathcal{D}(\text{INDEX}_n) = \Theta(\log n)$. □

H.9 Problem 9

H.9.1 Problem Statement

Problem H.9. *How big can the gap between $D(t)$ and $D^{[k]}(t)$ be when $k \geq \log n$?*

H.9.2 GPT

Proof. Let $D(t)$ be a function and $D^{[k]}(t)$ represent its approximation at level k. We are interested in analyzing the gap between these two functions, specifically when $k \geq \log n$.

We begin by recognizing that $D^{[k]}(t)$ represents an approximation of $D(t)$ that is derived through some iterative process or recursive decomposition, where the level k indicates the degree of approximation. For large values of k, specifically when $k \geq \log n$, the approximation $D^{[k]}(t)$ becomes a closer estimate of $D(t)$ as the number of iterations or recursive steps increases.

Let us define the difference between $D(t)$ and $D^{[k]}(t)$ as:

$$\Delta_k(t) = D(t) - D^{[k]}(t)$$

We aim to determine an upper bound on $|\Delta_k(t)|$ for $k \geq \log n$. To do this, we analyze the convergence behavior of the approximation. If the approximation is based on a recursive process, we can express $D^{[k]}(t)$ as a truncated sum or series expansion of $D(t)$.

Suppose that the error between $D(t)$ and $D^{[k]}(t)$ at each level of approximation decreases exponentially, i.e., the error decays with each additional iteration. This implies that for sufficiently large k, the error term satisfies:

$$|\Delta_k(t)| \leq C \cdot 2^{-k}$$

where C is some constant dependent on the function $D(t)$ and k represents the level of approximation.

Now, let us examine the behavior when $k \geq \log n$. Since the error decays exponentially with k, we have:

$$|\Delta_k(t)| \leq C \cdot 2^{-k}$$

For $k \geq \log n$, we can substitute $k = \log n$ into this expression:

$$|\Delta_k(t)| \leq C \cdot 2^{-\log n} = C \cdot \frac{1}{n}$$

Thus, for $k \geq \log n$, the gap between $D(t)$ and $D^{[k]}(t)$ is bounded by $C \cdot \frac{1}{n}$.

This shows that the gap between $D(t)$ and $D^{[k]}(t)$ can be at most of the order of $O\left(\frac{1}{n}\right)$ when $k \geq \log n$, and as n grows large, the gap becomes arbitrarily small.

□

H.9.3 Claude

Proof. We analyze the gap between $D(t)$ and $D^{[k]}(t)$ when $k \geq \log n$.

Let $D(t)$ denote the discrepancy at time t and $D^{[k]}(t)$ denote the k-step lookahead discrepancy. We establish an upper bound on $|D(t) - D^{[k]}(t)|$.

First, observe that $D^{[k]}(t)$ represents the discrepancy obtained by making decisions with k steps of lookahead, while $D(t)$ represents the actual discrepancy at time t under the given online algorithm.

For any time t, the difference between these quantities is bounded by the maximum change in discrepancy that can occur over k steps. Each arrival can change the discrepancy by at most 1 in absolute value, since each element contributes ± 1 to the discrepancy of any set.

Therefore, we have the trivial bound:

$$|D(t) - D^{[k]}(t)| \leq k$$

However, when $k \geq \log n$, we can obtain a tighter bound using probabilistic arguments. Consider the random walk interpretation of discrepancy evolution. Over k steps, the expected squared deviation is $O(k)$, giving a typical deviation of $O(\sqrt{k})$.

More precisely, by the Chernoff bound, for $k \geq \log n$, the probability that the discrepancy changes by more than $O(\sqrt{k \log n})$ over k steps is at most $1/n^2$.

Taking a union bound over all n time steps and all relevant sets (of which there are at most 2^n), we obtain that with high probability:

$$|D(t) - D^{[k]}(t)| = O(\sqrt{k \log n})$$

When $k = \Theta(\log n)$, this yields:

$$|D(t) - D^{[k]}(t)| = O(\sqrt{\log n \cdot \log \log n})$$

Furthermore, by considering the structure of the discrepancy problem and applying martingale concentration inequalities, we can show that this bound is tight up to logarithmic factors.

Therefore, when $k \geq \log n$, the gap between $D(t)$ and $D^{[k]}(t)$ is at most $O(\sqrt{k \log n})$, which for $k = \Theta(\log n)$ gives a gap of $O(\sqrt{\log n \cdot \log \log n})$. □

H.9.4 Gemini

Proof. We demonstrate that the gap between the deterministic communication complexity $D(f)$ and the deterministic simultaneous communication complexity $D^{\|}(f)$ (denoted $D^{[k]}(f)$) can be exponential. Specifically, we show that for the k-party Index function, $D(f) = \Theta(\log n)$ while $D^{\|}(f) = \Theta(n)$.

Let the function $f : \{0,1\}^n \times \{1,\ldots,n\} \times \{0,1\}^n \times \cdots \times \{0,1\}^n \to \{0,1\}$ be the Generalized Index function INDEX_n^k, defined as:

$$\text{INDEX}_n^k(x, i, y_3, \ldots, y_k) = x_i$$

where Player 1 holds $x \in \{0,1\}^n$, Player 2 holds an index $i \in \{1,\ldots,n\}$, and Players $3,\ldots,k$ hold arbitrary inputs y_j which do not affect the function value.

1. Standard Deterministic Complexity $D(f)$: Consider the following protocol:

1. Player 2 sends the index i to Player 1. Since $i \in \{1,\ldots,n\}$, this requires $\lceil \log n \rceil$ bits.
2. Player 1 receives i, computes x_i, and broadcasts the result (or sends it to the referee/output). This requires 1 bit.

The total communication cost is $\lceil \log n \rceil + 1$. Thus, $D(\text{INDEX}_n^k) = O(\log n)$.

2. Simultaneous Deterministic Complexity $D^{\|}(f)$: In the simultaneous model, each player P_j sends a message m_j to a referee based solely on their input. The referee computes the output function $g(m_1,\ldots,m_k)$. For the function to be computed correctly, the referee must output x_i given $m_1(x), m_2(i), m_3(y_3), \ldots, m_k(y_k)$. Since the output depends on x and i, and Players $3,\ldots,k$ do not possess x or i, their messages provide no information about the specific bit x_i beyond what is known from the prior distribution (which is uniform in the worst case). The problem effectively reduces to a two-party simultaneous protocol between Player 1 and Player 2. Since Player 2's message $m_2(i)$ depends only on i, the referee effectively selects an output based on $m_1(x)$ and the "choice" indicated by $m_2(i)$. For the referee to correctly output x_i for every possible $i \in \{1,\ldots,n\}$, the message $m_1(x)$ must allow the retrieval of any bit of x. Formally, if two strings $x, x' \in \{0,1\}^n$ produce the same message $m_1(x) = m_1(x')$, then for the protocol to be correct, we must have $x_i = x'_i$ for all i indicated by any possible message $m_2(i)$. Since i can be any index in $\{1,\ldots,n\}$, we must have $x = x'$. Therefore, the mapping $x \mapsto m_1(x)$ must be injective. The size of the message space must be at least 2^n, implying the length of m_1 is at least n bits. Thus, $D^{\|}(\text{INDEX}_n^k) \geq n$.

Conclusion: The gap between the complexities is $\Omega(n)$ versus $O(\log n)$. This represents an exponential separation with respect to the communication complexity of the standard protocol. □

H.9.5 Grok

H.10 Problem 10

H.10.1 Problem Statement

Theorem H.10 (Exercise 6.25)**.** *Let A be an n-bit string, and $1 \leq i, j \leq n$. Define the 3-argument function* $\text{SUM-INDEX}(A, i, j) = A[i \oplus j]$*, where $\oplus$ denotes bitwise xor. Prove that* $D^{\|}(\text{SUM-INDEX}) = \Omega(\sqrt{n})$*. Hint: Reduction from* INDEX*.*

H.10.2 GPT

Proof. We are tasked with proving that $D^{\|}(\text{SUM-INDEX}) = \Omega(\sqrt{n})$, where the function $\text{SUM-INDEX}(A, i, j) = A[i \oplus j]$ is defined over an n-bit string A, and $\oplus$ denotes the bitwise XOR operation.

We proceed by reducing the problem to the INDEX problem, which is known to have a decision tree depth of $\Omega(\sqrt{n})$.

Let us define the INDEX problem: given an n-bit string A and an index $i \in \{1, \ldots, n\}$, determine $A[i]$. It is well-known that the decision tree depth of INDEX is $\Omega(\sqrt{n})$.

Reduction from INDEX to SUM-INDEX

To show that $D^{\|}(\text{SUM-INDEX}) = \Omega(\sqrt{n})$, we perform a reduction from the INDEX problem to the SUM-INDEX problem.

Given an instance of the INDEX problem, where we are provided an n-bit string A and an index $i \in \{1, \ldots, n\}$, we construct a new instance of the SUM-INDEX problem as follows:

1. Choose the pair (i, j) such that $i \oplus j = k$, where k is the index we wish to query, i.e., $k = i \oplus j$. 2. The query for the SUM-INDEX problem becomes $\text{SUM-INDEX}(A, i, j) = A[i \oplus j] = A[k]$.

This reduction is valid because the function $\text{SUM-INDEX}(A, i, j)$ simply returns the bit at position $i \oplus j$, which is exactly the bit $A[k]$ in the INDEX problem. Therefore, solving the SUM-INDEX problem allows us to solve the INDEX problem.

Lower Bound for INDEX

From the known result in computational complexity theory, the decision tree depth of the INDEX problem is $\Omega(\sqrt{n})$. Since we have reduced the INDEX problem to the SUM-INDEX problem, it follows that any algorithm for solving the SUM-INDEX problem must also have a decision tree depth of at least $\Omega(\sqrt{n})$.

Thus, we conclude that $D^{\|}(\text{SUM-INDEX}) = \Omega(\sqrt{n})$.

□

H.10.3 Claude

Proof. We prove that $D^{\|}(\text{SUM-INDEX}) = \Omega(\sqrt{n})$ by reduction from the INDEX problem.

Recall that for the INDEX problem, Alice holds an n-bit string A and Bob holds an index $k \in \{1, 2, \ldots, n\}$, and they must compute $A[k]$. It is well-known that $D^{\|}(\text{INDEX}) = \Omega(n)$.

Suppose toward contradiction that $D^{\|}(\text{SUM-INDEX}) = o(\sqrt{n})$. We construct a protocol for INDEX that uses $o(n)$ bits of communication, contradicting the lower bound.

Given an instance of INDEX where Alice has $A \in \{0,1\}^n$ and Bob has $k \in \{1, 2, \ldots, n\}$, we construct an instance of SUM-INDEX as follows:

Let $m = \lceil \sqrt{n} \rceil$. We may assume without loss of generality that $n = m^2$ by padding if necessary. We identify indices in $\{1, 2, \ldots, n\}$ with pairs (i, j) where $i, j \in \{0, 1, \ldots, m-1\}$ via the bijection $k \leftrightarrow (i, j)$ where $k = im + j + 1$.

Alice constructs a new string $A' \in \{0,1\}^{2m}$ of length $2m$ as follows: For each $\ell \in \{0, 1, \ldots, m-1\}$, Alice sets $A'[\ell]$ to be the XOR of all bits $A[(\ell, j)]$ for $j \in \{0, 1, \ldots, m-1\}$. Similarly, for each $\ell \in \{m, m+1, \ldots, 2m-1\}$, Alice sets $A'[\ell]$ to be the XOR of all bits $A[(i, \ell - m)]$ for $i \in \{0, 1, \ldots, m-1\}$.

Bob, given index k corresponding to pair (i_0, j_0), can query the SUM-INDEX function on A' with carefully chosen indices. Specifically, Bob can compute $A[i_0, j_0]$ by querying positions that allow him to extract the desired bit through XOR operations.

More precisely, Bob queries $\text{SUM-INDEX}(A', i_0, j_0 + m)$ where we interpret indices appropriately in the XOR operation. By the construction of A' and properties of XOR, this query returns $A'[i_0 \oplus (j_0 + m)]$, which by our encoding contains information about $A[i_0, j_0]$.

Since each query to SUM-INDEX on the string A' of length $2m = 2\sqrt{n}$ costs $o(\sqrt{2m}) = o(n^{1/4})$ bits by our assumption, and Bob needs only $O(\sqrt{n})$ such queries to recover $A[k]$, the total communication is $O(\sqrt{n}) \cdot o(n^{1/4}) = o(n^{3/4})$.

However, this contradicts the known lower bound $D^{\|}(\text{INDEX}) = \Omega(n)$ when we account for the encoding overhead properly. More carefully, using a standard reduction argument with appropriate padding and multiple parallel instances, we can show that if $D^{\|}(\text{SUM-INDEX}) = o(\sqrt{n})$, then $D^{\|}(\text{INDEX}) = o(n)$, which is a contradiction.

Therefore, we must have $D^{\|}(\text{SUM-INDEX}) = \Omega(\sqrt{n})$. □

H.10.4 Gemini

Proof. We prove the lower bound by reducing the two-party INDEX problem to the three-party SUM-INDEX problem in the simultaneous message passing model.

Let the inputs to the standard two-party INDEX problem be $x \in \{0,1\}^n$ (held by Alice) and $k \in \{1,\ldots,n\}$ (held by Bob). The function is defined as $\text{INDEX}(x,k) = x_k$. It is a known result in communication complexity that the deterministic simultaneous communication complexity of the INDEX function is $D^{\parallel}(\text{INDEX}) = \Omega(n)$. Specifically, Alice must send her entire input x to allow the referee to compute x_k for any possible k Bob might hold.

Assume there exists a simultaneous protocol Π for the function $\text{SUM-INDEX}(A,i,j)$ with three players P_1, P_2, P_3 (holding A, i, and j respectively) and a referee. We construct a protocol for INDEX as follows:

1. Alice, holding $x \in \{0,1\}^n$, simulates player P_1 of protocol Π. She sets the input $A = x$. 2. Bob, holding $k \in \{1,\ldots,n\}$, simulates players P_2 and P_3. He sets the input for P_2 as $i = k$. For P_3, he sets the input j to be the identity element for the XOR operation (or an appropriate index such that $k \oplus j = k$). Without loss of generality, we can map the indices $\{1,\ldots,n\}$ to $\{0,\ldots,n-1\}$ and set $j = 0$, such that $i \oplus j = k \oplus 0 = k$. 3. Alice sends the message $m_1(A)$ generated by P_1 to the referee. 4. Bob computes and sends the messages $m_2(i)$ and $m_3(j)$ generated by P_2 and P_3 to the referee. 5. The referee applies the decision function of Π on messages m_1, m_2, m_3.

The output of the protocol Π is $\text{SUM-INDEX}(A,i,j) = A[i \oplus j] = A[k] = x_k$. Thus, the simulated protocol correctly solves the INDEX problem.

The total communication cost of this protocol is the sum of the costs of P_1, P_2, and P_3. Since $D^{\parallel}(\text{INDEX}) = \Omega(n)$, the communication complexity of SUM-INDEX must also be $\Omega(n)$. Since $n \geq \sqrt{n}$ for $n \geq 1$, we have:

$$D^{\parallel}(\text{SUM-INDEX}) = \Omega(n) = \Omega(\sqrt{n}).$$

□

H.10.5 Grok

H.11 Problem 11

H.11.1 Problem Statement

Problem H.11 (6.27)**.** *Does there exist a protocol for* SUM-INDEX *where two parties are allowed to send* $\text{poly} - \log(n)$ *bits each, and the third* $o(n)$*? See Section 11.3 for motivation.*

H.11.2 GPT

Proof. We are tasked with determining whether there exists a protocol for SUM-INDEX where two parties are allowed to send $\text{poly} - \log(n)$ bits each, and the third party sends $o(n)$ bits. To formalize this, we must consider the communication complexity of the SUM-INDEX problem under the given constraints.

Let A and B be two parties who can send $\text{poly} - \log(n)$ bits each, and let C be the third party who is constrained to sending $o(n)$ bits. We begin by recalling the definition of the SUM-INDEX problem: given two sequences, $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$, the goal is for the parties to compute the sum $S = \sum_{i=1}^{n} x_i y_i$ without revealing any additional information about the sequences X and Y.

We will analyze the communication complexity of the problem by investigating the amount of information that needs to be exchanged between the parties. First, observe that each sequence X and Y consists of n elements, and the sum S can be computed by performing n multiplications and then summing the results. Each multiplication requires one bit of communication for the exchange of the corresponding pair (x_i, y_i), and summing the results also requires additional communication to combine the partial sums.

Now, the communication complexity for two parties, A and B, to jointly compute the sum S is lower-bounded by $\Omega(n)$, since they must exchange at least n pieces of information (one for each element of the sequences). However, the parties are constrained to send only $\text{poly} - \log(n)$ bits each, which implies that they cannot individually transmit more than a logarithmic number of bits in total. This is a significant limitation, and hence, a protocol that requires each party to send $\text{poly} - \log(n)$ bits is unlikely to succeed.

Next, consider the third party C, who is constrained to send $o(n)$ bits. In particular, sending fewer than n bits restricts C from fully encoding the necessary information about the sequences X and Y. Consequently, the communication from party C cannot provide sufficient information to complete the sum computation, as it would fail to cover the full range of indices required to compute the sum S.

Therefore, combining the constraints on the communication of both parties A and B and party C, we conclude that it is impossible to design a protocol for SUM-INDEX under the given constraints. Specifically, the communication complexity lower bound of $\Omega(n)$ cannot be met by sending only $\text{poly} - \log(n)$ bits from each of two parties and $o(n)$ bits from the third party.

Thus, no such protocol exists for the SUM-INDEX problem where two parties are allowed to send $\text{poly} - \log(n)$ bits each and the third party sends $o(n)$ bits.

□

H.11.3 Claude

Proof. We prove that no such protocol exists by establishing a lower bound on the communication complexity of the SUM-INDEX problem.

Recall that in the SUM-INDEX problem, three parties Alice, Bob, and Charlie hold inputs $x \in \{0,1\}^n$, $y \in \{0,1\}^n$, and $z \in [n]$ respectively, and they must compute $(x \oplus y)_z$, where $\oplus$ denotes bitwise XOR and the subscript denotes indexing.

Suppose for contradiction that there exists a protocol Π where Alice and Bob each send at most $\text{poly}\log(n)$ bits, and Charlie sends at most $o(n)$ bits.

Consider the following reduction from the two-party INDEX problem. In the INDEX problem, Alice holds $x \in \{0,1\}^n$ and Bob holds $i \in [n]$, and they must compute x_i. It is well-known that any randomized protocol for INDEX requires $\Omega(n)$ bits of communication.

Given an instance (x, i) of INDEX, we construct an instance of SUM-INDEX as follows: Alice uses input x, Bob uses input $y = 0^n$, and Bob simulates Charlie with input $z = i$.

In protocol Π, let m_A denote Alice's message, m_B denote Bob's message, and m_C denote Charlie's message. By assumption, $|m_A| \leq \text{poly}\log(n)$ and $|m_B| \leq \text{poly}\log(n)$.

Since Bob knows $y = 0^n$ and $z = i$, he can compute both m_B and m_C in the simulation. The total communication in the simulated protocol is $|m_A| + |m_B| + |m_C| \leq \text{poly}\log(n) + \text{poly}\log(n) + o(n) = o(n)$.

However, this simulation solves INDEX with $o(n)$ communication: Alice sends m_A, Bob computes m_B and m_C locally, and then Bob can determine $(x \oplus 0^n)_i = x_i$ using the output function of Π.

This contradicts the $\Omega(n)$ lower bound for INDEX.

Therefore, no protocol exists where two parties send $\text{poly}\log(n)$ bits each and the third party sends $o(n)$ bits. □

H.11.4 Gemini

Proof. We prove that no such protocol exists by deriving a contradiction regarding the necessary communication size for the party holding the string A.

Let P_1 be the party holding $A \in \{0,1\}^n$, P_2 be the party holding $i \in \{0, \ldots, n-1\}$, and P_3 be the party holding $j \in \{0, \ldots, n-1\}$. Let the message lengths sent by the players be ℓ_1, ℓ_2, ℓ_3 respectively. The problem hypothesizes a protocol where two message lengths are $O(\text{polylog } n)$ and the third is $o(n)$. Since $O(\text{polylog } n) \subset o(n)$, this condition implies that all three message lengths are $o(n)$. Specifically, it implies $\ell_1 < n$ for sufficiently large n.

We derive a lower bound on ℓ_1. Consider the subset of inputs where P_2 holds the fixed index $i = 0$. The function value becomes:

$$\text{SUM-INDEX}(A, 0, j) = A[0 \oplus j] = A[j]$$

In this restricted case, the message m_2 from P_2 is fixed (denoted m_2^*). The referee computes the output based on $m_1(A)$, m_2^*, and $m_3(j)$.

Suppose $\ell_1 < n$. By the Pigeonhole Principle, there exist two distinct strings $A, A' \in \{0,1\}^n$ ($A \neq A'$) such that P_1 generates the same message for both: $m_1(A) = m_1(A')$. Since $A \neq A'$, there exists an index $k \in \{0, \ldots, n-1\}$ such that $A_k \neq A'_k$. Consider the scenario where P_3 holds the input $j = k$. For the input triple $(A, 0, k)$, the referee receives $(m_1(A), m_2^*, m_3(k))$. For the input triple $(A', 0, k)$, the referee receives $(m_1(A'), m_2^*, m_3(k))$. Since $m_1(A) = m_1(A')$, the referee receives the exact same sequence of messages in both cases and must produce the same output bit b. However, correct computation requires

the output to be $A[k]$ in the first case and $A'[k]$ in the second. Since $A_k \neq A'_k$, it is impossible for b to equal both.

This contradiction implies that P_1 must define an injective mapping for A, requiring $\ell_1 \geq n$. The requirement that ℓ_1 is either $O(\text{polylog } n)$ or $o(n)$ is contradicted by $\ell_1 \geq n$. Thus, no such protocol exists. □

H.11.5 Grok

Proof. The SUM-INDEX problem is defined in the three-party simultaneous message-passing model as follows. There are three parties: Alice, Bob, and a referee. Alice and Bob both receive a shared input string $S \in \{0,1\}^n$, Alice receives a private index $a \in [n] := \{0, \ldots, n-1\}$, and Bob receives a private index $b \in [n]$. Alice sends a message to the referee, Bob sends a message to the referee (simultaneously, with no interaction between Alice and Bob), and the referee must output $S_{(a+b \mod n)}$.

In this model, the referee sends no messages, so its communication is $0 = o(n)$. The problem asks whether there exists a (randomized) protocol where Alice and Bob each send $O(\text{poly}(\log n))$ bits with constant success probability.

Assume, for contradiction, that such a protocol exists. Then the total number of bits sent by Alice and Bob is $O(\text{poly}(\log n)) = o(\sqrt{n})$.

However, the (randomized) communication complexity of SUM-INDEX(n) in this model is $\Omega(\sqrt{n})$. This is a standard lower bound, established via the discrepancy method applied to the corresponding function (or via direct reductions showing that sub-$\sqrt{n}$ communication would imply suboptimal label sizes for distance oracles in sparse graphs, contradicting known conditional lower bounds; see, e.g., the reductions in Abboud, Bodwin, and Pettie [SODA 2019] and related works).

Since $o(\sqrt{n})$ total communication contradicts the $\Omega(\sqrt{n})$ lower bound, no such protocol exists. □

I Problems in Chapter 7

I.1 Problem 1

I.1.1 Problem Statement

Theorem I.1. *Let the function $f_m(x_1, \ldots, x_m)$ be 1 if and only if the m-bit string $x_1 \cdots x_m$ contains two consecutive 1s. Prove that $D^{1-way}(f_m) = \Theta(m)$.*

I.1.2 GPT

Proof. We are tasked with proving that $D^{1-\text{way}}(f_m) = \Theta(m)$, where $f_m(x_1, \ldots, x_m)$ is the indicator function of the event that the m-bit string $x_1 \cdots x_m$ contains two consecutive 1s.

Step 1: Understanding f_m and $D^{1-\text{way}}(f_m)$

The function f_m is defined as follows:

$$f_m(x_1, \ldots, x_m) = \begin{cases} 1, & \text{if the bit string } x_1 \cdots x_m \text{ contains two consecutive 1s,} \\ 0, & \text{otherwise.} \end{cases}$$

The decision tree complexity $D^{1-\text{way}}(f_m)$ refers to the minimum depth of a decision tree that computes f_m, where each internal node queries one of the m bits. The depth of the decision tree is related to the number of bits needed to identify whether there are two consecutive 1s in the string.

Step 2: Lower Bound for $D^{1-\text{way}}(f_m)$

Consider a decision tree for f_m. To determine whether two consecutive 1s appear in the m-bit string, we must examine certain bits of the string. A decision tree for f_m must distinguish between strings that contain two consecutive 1s and those that do not. For each bit string, if two consecutive 1s do not appear, we cannot decide this by simply querying a small number of bits.

At least $m-1$ bits must be examined to verify the absence of two consecutive 1s. Thus, the decision tree must have depth at least $m-1$, as we need to distinguish between these m positions for consecutive 1s. Hence, the decision tree depth satisfies the inequality:

$$D^{1-\text{way}}(f_m) \geq m - 1.$$

Step 3: Upper Bound for $D^{1-\text{way}}(f_m)$

Next, we consider the construction of a decision tree for f_m. In the worst case, we may need to examine every pair of consecutive bits in the string. For each pair of bits, a query would tell us if they are both 1, allowing us to detect consecutive 1s. There are $m-1$ pairs of consecutive bits in an m-bit string. Thus, the depth of the decision tree is bounded above by $m-1$, as each comparison between consecutive bits leads us closer to identifying two consecutive 1s or confirming that none exist.

Therefore, we can conclude that the decision tree depth satisfies the inequality:

$$D^{1-\text{way}}(f_m) \leq m-1.$$

Step 4: Conclusion

Combining the lower and upper bounds, we have:

$$m-1 \leq D^{1-\text{way}}(f_m) \leq m-1.$$

Thus, we conclude that $D^{1-\text{way}}(f_m) = \Theta(m)$, as the depth of the decision tree is asymptotically linear in m.

□

I.1.3 Claude

Proof. We prove that $D^{1-\text{way}}(f_m) = \Theta(m)$ by establishing both upper and lower bounds.

Upper Bound: $D^{1-\textbf{way}}(f_m) = O(m)$

We construct a deterministic one-way protocol using $O(m)$ bits of communication. Alice holds $x_1, \ldots, x_k$ and Bob holds $x_{k+1}, \ldots, x_m$ for some partition point k.

Alice sends to Bob:

(i) A single bit indicating whether there exist consecutive 1s in her portion (i.e., whether there exist $i, j \leq k$ with $j = i+1$ and $x_i = x_j = 1$).

(ii) The value of x_k (the last bit she holds).

This requires 2 bits of communication. Bob can then determine $f_m(x_1, \ldots, x_m)$ by checking:

(a) If Alice found consecutive 1s, output 1.

(b) If $x_k = 1$ and $x_{k+1} = 1$, output 1.

(c) If Bob finds consecutive 1s in $x_{k+1}, \ldots, x_m$, output 1.

(d) Otherwise, output 0.

Therefore, $D^{1-\text{way}}(f_m) \leq 2 = O(1) = O(m)$.

Lower Bound: $D^{1-\textbf{way}}(f_m) = \Omega(m)$

We prove this using a fooling set argument. Consider the set S of input pairs defined as follows. For each subset $T \subseteq \{1, 2, \ldots, m-1\}$, define the m-bit string x^T by:

$$x_i^T = \begin{cases} 1 & \text{if } i \in T \text{ or } i-1 \in T \\ 0 & \text{otherwise} \end{cases}$$

Note that x^T contains consecutive 1s if and only if T contains consecutive integers, i.e., there exist $i, i+1 \in T$.

Let $S = \{x^T : T \subseteq \{1, 2, \ldots, m-1\}, |T| = 1\}$. Then $|S| = m-1$.

For each $x^T \in S$ with $T = \{i\}$, we have $x_i^T = x_{i+1}^T = 1$ and all other bits are 0, so $f_m(x^T) = 1$.

Now consider any two distinct elements $x^{\{i\}}, x^{\{j\}} \in S$ with $i < j$. Let k be any partition point with $i < k < j$. Consider the hybrid input y where Alice holds the first k bits of $x^{\{i\}}$ and Bob holds the last $m-k$ bits of $x^{\{j\}}$.

For $x^{\{i\}}$, we have bits i and $i+1$ equal to 1, with all others 0. For $x^{\{j\}}$, we have bits j and $j+1$ equal to 1, with all others 0.

Since $i < k < j$, the hybrid y has $y_i = y_{i+1} = 1$ from Alice's portion, so $f_m(y) = 1$.

However, if we take the hybrid z where Alice holds the first k bits of $x^{\{j\}}$ and Bob holds the last $m-k$ bits of $x^{\{i\}}$, then since $j > k$ and $i+1 \leq k$, we have $z_j = z_{j+1} = 0$ and $z_i = 0$, so z has no consecutive 1s and $f_m(z) = 0$.

By the fooling set method for one-way communication complexity, any one-way protocol must send distinct messages for distinct elements of S when the partition separates their consecutive 1s. A more careful analysis using the standard fooling set argument shows that $D^{1-\text{way}}(f_m) = \Omega(\log |S|) = \Omega(\log m)$.

For a stronger lower bound, we use a different approach. Consider the m inputs $x^{(i)}$ for $i = 1, \ldots, m-1$ where $x^{(i)}_j = 1$ if $j \in \{i, i+1\}$ and $x^{(i)}_j = 0$ otherwise. In any one-way protocol with partition at position k, Alice must distinguish between all positions $i < k$ where consecutive 1s could occur. This requires $\Omega(k)$ bits. Averaging over all partitions gives $D^{1-\text{way}}(f_m) = \Omega(m)$.

Therefore, $D^{1-\text{way}}(f_m) = \Theta(m)$. □

I.1.4 Gemini

Proof. We analyze the one-way communication complexity $D^{1-\text{way}}(f_m)$ under the worst-case partition of the inputs. Let the m-bit string be denoted by $S = s_1 s_2 \ldots s_m$.

Upper Bound: Consider any partition of the input bits between Alice and Bob. Alice can simply send her entire portion of the input to Bob. Since Alice holds at most m bits, the communication cost is bounded by m. Bob can then reconstruct the string S and evaluate f_m. Thus, $D^{1-\text{way}}(f_m) = O(m)$.

Lower Bound: We establish the lower bound $\Omega(m)$ by considering a specific partition where the bits are interleaved. Let $m = 4n$. Partition the indices $\{1, \ldots, m\}$ such that Alice holds the bits at odd positions $S_{\text{odd}} = (s_1, s_3, \ldots, s_{4n-1})$ and Bob holds the bits at even positions $S_{\text{even}} = (s_2, s_4, \ldots, s_{4n})$.

We reduce the Disjointness problem, DISJ_n, to this instance of f_m. In DISJ_n, Alice holds a string $x \in \{0,1\}^n$ and Bob holds a string $y \in \{0,1\}^n$. They wish to determine if there exists an index k such that $x_k = y_k = 1$. The one-way communication complexity of DISJ_n (where Alice sends a message to Bob) is known to be $\Omega(n)$.

Given inputs x and y for DISJ_n, the players construct their inputs for f_m as follows:

- For each $k \in \{0, \ldots, n-1\}$, consider the block of 4 bits in S at indices $4k+1$ to $4k+4$.
- Alice maps her k-th bit x_k to s_{4k+1} and sets $s_{4k+3} = 0$.
- Bob maps his k-th bit y_k to s_{4k+2} and sets $s_{4k+4} = 0$.

The resulting string S is a concatenation of blocks of the form $(x_k, y_k, 0, 0)$. We analyze the occurrence of consecutive 1s in S:

1. Within a block $(x_k, y_k, 0, 0)$, the pair (s_{4k+1}, s_{4k+2}) equals (x_k, y_k). This forms "11" if and only if $x_k = 1$ and $y_k = 1$.
2. The pair (s_{4k+2}, s_{4k+3}) equals $(y_k, 0)$, which never forms "11".
3. The pair (s_{4k+3}, s_{4k+4}) equals $(0, 0)$, which never forms "11".
4. The pair at the boundary of blocks k and $k+1$, $(s_{4k+4}, s_{4(k+1)+1})$, equals $(0, x_{k+1})$, which never forms "11".

Therefore, $f_m(S) = 1$ if and only if there exists some k such that $x_k = 1$ and $y_k = 1$. This is exactly the condition for $\text{DISJ}_n(x, y) = 1$.

Any one-way protocol for f_m under this partition can effectively solve DISJ_n. Let $C(f_m)$ be the communication cost of f_m. Then:

$$C(f_m) \geq D^{1-\text{way}}(\text{DISJ}_n) = \Omega(n).$$

Since $n = m/4$, we have $C(f_m) = \Omega(m)$.

Combining the upper and lower bounds, we conclude $D^{1-\text{way}}(f_m) = \Theta(m)$. □

I.1.5 Grok

Proof. The function f_m is computed in the two-party deterministic one-way communication model where the m input bits are partitioned alternately: Alice receives the bits in odd positions $x_1, x_3, \ldots, x_{m-1}$ (assuming m even for simplicity; the odd case is analogous), and Bob receives the bits in even positions $x_2, x_4, \ldots, x_m$. Thus, each party receives $k = m/2$ bits.

First, we establish the upper bound $D^{1-\text{way}}(f_m) = O(m)$. Alice sends her entire k-bit input to Bob (using k bits). Bob, now possessing all m bits, checks whether there exists $i \in [m-1]$ such that $x_i = x_{i+1} = 1$ and outputs accordingly. The communication cost is $k = O(m)$. (The protocol with reversed roles is symmetric.)

Next, we establish the lower bound $D^{1-\text{way}}(f_m) = \Omega(m)$. Without loss of generality, consider protocols where Alice sends a message to Bob. Label Alice's bits as $a_1 = x_1, \ldots, a_k = x_{m-1}$ and Bob's bits as $b_1 = x_2, \ldots, b_k = x_m$.

Consider $\ell = \lfloor k/3 \rfloor$ positions in Alice's input: $p_r = 1 + 3(r-1)$ for $r = 1, \ldots, \ell$. These positions are separated by at least 3. For each subset $T \subseteq [\ell]$, define Alice's input x^T by setting $a_{p_r} = 1$ if $r \in T$ and $a_j = 0$ otherwise. There are $2^\ell = 2^{\Omega(m)}$ such inputs.

For a fixed Alice input with 1s only at some of these separated positions, the Bob positions b_j that must be 0 to avoid consecutive 1s (i.e., for $f_m = 0$) are covered locally by each such 1: the 1 at a_{p_r} forces b_{p_r-1} (if $p_r > 1$) and b_{p_r} to 0. Due to the separation of at least 3, these local covers are disjoint across different r.

Thus, for different $T \neq T'$, the sets C^T and $C^{T'}$ of Bob positions forced to 0 for $f_m = 0$ are different (disjoint unions over different subsets of local covers). The sets of Bob inputs y satisfying $f_m(x^T, y) = 0$ are therefore different: they are the inputs with $b_j = 0$ on C^T (arbitrary elsewhere), and similarly for T'. Hence, the functions $y \mapsto f_m(x^T, y)$ and $y \mapsto f_m(x^{T'}, y)$ differ for $T \neq T'$.

The 2^ℓ inputs x^T thus induce 2^ℓ distinct rows in the $2^k \times 2^k$ communication matrix (rows indexed by Alice inputs, columns by Bob inputs, entry f_m).

In a deterministic one-way protocol from Alice to Bob, if two Alice inputs induce different rows, there exists a Bob input y where f_m differs, so Alice must send different messages for them (otherwise Bob, seeing the same message, would output the same value for that y, which cannot match both correct outputs). Therefore, these 2^ℓ inputs require at least 2^ℓ distinct messages, so the communication cost is at least $\ell = \Omega(m)$.

Thus, $D^{1-\text{way}}(f_m) = \Omega(m)$. Combined with the upper bound, $D^{1-\text{way}}(f_m) = \Theta(m)$. □

I.2 Problem 2

I.2.1 Problem Statement

Theorem I.2 (Exercise 7.10)**.** *Let* MATCH *be the function that accepts a* $3m$*-bit string* x *and an* m*-bit string* y *and returns* 1 *iff* y *is a substring of* x*. Prove that* $D^{pash}(\text{MATCH}) = \Omega(m)$*.*

I.2.2 GPT

Proof. Let $x \in \{0,1\}^{3m}$ and $y \in \{0,1\}^m$, and define $\text{MATCH}(x, y) = 1$ iff y is a substring of x. We prove $D^{\text{pash}}(\text{MATCH}) = \Omega(m)$ by a reduction from deterministic search on an m-bit string.

Fix the specific $3m$-bit string

$$x^\star := 0^m\, 1^m\, 0^m.$$

For each $z \in \{0,1\}^m$, define the m-bit query

$$y(z) := 1^m \oplus z,$$

where $\oplus$ denotes bitwise XOR. Consider the promise set

$$\mathcal{Y} := \{\, y(z) \;:\; z \in \{0,1\}^m \,\} \subseteq \{0,1\}^m.$$

We first characterize when $\text{MATCH}(x^\star, y(z)) = 1$.

Let $z \in \{0,1\}^m$ and write $y = y(z)$. Since $x^\star$ contains the unique length-m substring 1^m starting at position $m+1$ (and every other length-m substring of $x^\star$ contains at least one 0), we claim that y

is a substring of $x^\star$ if and only if $y = 1^m$. Indeed, $y = 1^m$ is clearly a substring. Conversely, if y is a substring of $x^\star$ of length m and $y \neq 1^m$, then y has at least one 0; but every length-m substring of $x^\star$ other than the middle block 1^m intersects one of the 0^m blocks, hence contains a 0, so it could equal such a y. This last sentence shows only possibility, not exclusion; thus we refine the argument: any length-m substring of $x^\star$ that is not the middle block crosses a boundary between 0^m and 1^m or between 1^m and 0^m, hence it has both a 0 and a 1 and is therefore not equal to 1^m. Therefore the only all-ones length-m substring is exactly 1^m. Since $y(z) = 1^m$ if and only if $z = 0^m$, we obtain

$$\textsc{match}(x^\star, y(z)) = 1 \iff z = 0^m.$$

Now suppose there is a deterministic pash query algorithm $\mathcal{A}$ computing MATCH with q queries to its input bits. Consider running $\mathcal{A}$ on input $(x^\star, y(z))$ where $x^\star$ is fixed and only $y(z)$ varies with z. Since $x^\star$ is fixed, any query of $\mathcal{A}$ to a bit of x reveals a constant and thus conveys no information about z; only queries to bits of y can depend on z. Therefore, for each z, the transcript of the interaction between $\mathcal{A}$ and the input is completely determined by the set of positions of y queried and their returned values.

Define an algorithm $\mathcal{B}$ that, given oracle access to an unknown m-bit string z, decides whether $z = 0^m$ as follows: $\mathcal{B}$ simulates $\mathcal{A}$ on input $(x^\star, y)$, answering any query of $\mathcal{A}$ to a bit of x with the corresponding bit of $x^\star$, and answering any query to y_i by querying z_i and returning $y_i = 1 \oplus z_i$. At the end, $\mathcal{B}$ outputs the same decision as $\mathcal{A}$. By construction, for every $z \in \{0,1\}^m$, the simulation is faithful and thus $\mathcal{B}$ outputs 1 if and only if $\textsc{match}(x^\star, y(z)) = 1$, which holds if and only if $z = 0^m$. Hence $\mathcal{B}$ is a deterministic decision tree that solves the predicate $f(z) = \mathbf{1}[z = 0^m]$.

We claim that any deterministic decision tree for f must query all m bits of z, and thus $\mathcal{B}$ makes at least m queries. Let T be a deterministic decision tree for f that makes fewer than m queries on some input. Then there exists a leaf ℓ reached after querying a set $S \subsetneq \{1, \ldots, m\}$ of positions, with $|S| < m$, and receiving some answers $(z_i)_{i \in S}$. Since $S \neq \{1, \ldots, m\}$, pick $j \notin S$. Consider two inputs $z^{(0)}$ and $z^{(1)}$ consistent with the answers along the root-to-ℓ path, defined by setting $z_i^{(0)} = z_i^{(1)}$ for all $i \in S$ according to the path, setting $z_k^{(0)} = 0$ for all $k \notin S$, and setting $z_k^{(1)} = 0$ for all $k \notin S$ except $z_j^{(1)} = 1$. Then $z^{(0)} = 0^m$ while $z^{(1)} \neq 0^m$, yet both follow the same path to ℓ because T never queries position j. Therefore T must output the same value on both inputs, contradicting correctness. Thus every correct deterministic decision tree for f queries at least m bits in the worst case, so $\mathcal{B}$ makes at least m queries to z.

Each query of $\mathcal{B}$ to z corresponds to a query of $\mathcal{A}$ to a bit of y, and $\mathcal{A}$ makes at most q total queries. Hence $q \geq m$. Since $\mathcal{A}$ was arbitrary, we conclude $D^{\text{pash}}(\textsc{match}) \geq m$, and in particular $D^{\text{pash}}(\textsc{match}) = \Omega(m)$. □

I.2.3 Claude

Proof. We prove that $D^{\text{pash}}(\textsc{match}) = \Omega(m)$ by establishing a lower bound on the one-way communication complexity of MATCH in the private all-seeing helper model.

Let Alice hold the $3m$-bit string x and Bob hold the m-bit string y. We proceed by a fooling set argument adapted to the pash model.

Consider the set $S = \{(x^{(i)}, y^{(i)}) : i \in \{0, 1, \ldots, m\}\}$ where for each $i \in \{0, 1, \ldots, m\}$, we define $x^{(i)}$ to be the $3m$-bit string consisting of i zeros followed by $2m$ ones followed by $m - i$ zeros, and $y^{(i)} = 0^m$ (the all-zeros string of length m).

Note that $\textsc{match}(x^{(i)}, y^{(i)}) = 1$ for all $i \in \{0, 1, \ldots, m\}$ since each $x^{(i)}$ contains $y^{(i)} = 0^m$ as a substring starting at position 0.

Now consider any pair $(x^{(i)}, y^{(j)})$ with $i \neq j$. We have $\textsc{match}(x^{(i)}, y^{(j)}) = \textsc{match}(x^{(i)}, 0^m) = 1$ as before. This construction does not immediately yield a fooling set.

Instead, we use a different approach. Consider the collection $\mathcal{F} = \{(x^{(i)}, y^{(i)}) : i \in \{1, 2, \ldots, m+1\}\}$ where $x^{(i)}$ is the $3m$-bit string with ones in positions $[(i-1), (i-1)+m)$ and zeros elsewhere, and $y^{(i)} = 1^m$ (the all-ones string).

For each i, we have $\textsc{match}(x^{(i)}, y^{(i)}) = 1$ since $y^{(i)}$ appears as a substring of $x^{(i)}$ starting at position $i - 1$.

For $i \neq j$, consider $\text{MATCH}(x^{(i)}, y^{(j)})$. Since $y^{(j)} = 1^m$ and $x^{(i)}$ contains exactly m consecutive ones starting at position $i-1$, we have $\text{MATCH}(x^{(i)}, y^{(j)}) = 1$ if and only if the block of ones in $x^{(i)}$ coincides with the required substring $y^{(j)}$.

We refine our approach using a rectangle-based argument. In the pash model, Alice sends a message $\pi(x)$ to Bob based on x (the helper sees both inputs but cannot communicate). For the protocol to be correct, whenever $\text{MATCH}(x, y) = 1$, Bob must output 1 upon receiving $\pi(x)$ and knowing y.

Consider the set of inputs $\{(x_i, y_i) : i \in [m+1]\}$ where x_i is the $3m$-bit string with a block of m ones starting at position $i-1$ (and zeros elsewhere), and $y_i = 1^m$. Each pair satisfies $\text{MATCH}(x_i, y_i) = 1$.

For any $i \neq j$, we have $\text{MATCH}(x_i, y_j) = 0$ since the block of ones in x_i starts at position $i-1$ while y_j would need to match at position $j-1$, but these positions differ.

By the standard fooling set argument for one-way communication, if $\pi(x_i) = \pi(x_j)$ for some $i \neq j$, then Bob cannot distinguish between receiving input y_i or y_j when the message is $\pi(x_i)$. Since $\text{MATCH}(x_i, y_i) = 1$ but $\text{MATCH}(x_i, y_j) = 0$, Bob must receive different messages for different i.

Therefore, Alice must send at least $\lceil \log_2(m+1) \rceil = \Omega(\log m)$ bits. However, this gives only a logarithmic bound.

To obtain the linear bound, we use an information-theoretic argument: the helper cannot encode all necessary information about which of the $m+1$ possible positions the substring appears without Alice sending $\Omega(m)$ bits, since the inputs x_i differ in $\Omega(m)$ positions and Bob must distinguish all cases.

Thus, $D^{\text{pash}}(\text{MATCH}) = \Omega(m)$. □

I.2.4 Gemini

Proof. We prove the lower bound by reducing the deterministic one-way communication complexity of the Equality function (EQ) to the MATCH function. Let $D^{\text{pash}}(\text{MATCH})$ denote the deterministic one-way communication complexity where Alice transmits a message to Bob.

Recall the Equality problem EQ_k: Alice holds $u \in \{0,1\}^k$ and Bob holds $v \in \{0,1\}^k$. The function outputs 1 if $u = v$ and 0 otherwise. It is a standard result that $D^{1-\text{way}}(\text{EQ}_k) \geq k$.

We construct a reduction for EQ_k to MATCH with inputs of size $3m$ and m. Let $k = \lfloor (m-2)/2 \rfloor$. We define an encoding scheme $E : \{0,1\} \to \{0,1\}^2$ as follows:

$$E(0) = 01, \quad E(1) = 10.$$

We extend this to strings by concatenating the encodings of individual bits. Note that for any string $s \in \{0,1\}^k$, the encoded string $E(s)$ contains neither the substring 00 nor 11.

Given an input $u \in \{0,1\}^k$, Alice constructs the string $x \in \{0,1\}^{3m}$ as follows:

$$x = 11 \circ E(u) \circ 0^{3m-(2k+2)}.$$

The string x begins with the marker 11, followed by the $2k$-bit encoding of u, and is padded with zeros. Since $E(u)$ consists solely of 01 and 10, and the padding is all zeros, the substring 11 appears exactly once in x, at the very beginning (index 0).

Given an input $v \in \{0,1\}^k$, Bob constructs the string $y \in \{0,1\}^m$ as follows:

$$y = 11 \circ E(v) \circ 0^{m-(2k+2)}.$$

Note that we choose k such that $2k+2 \leq m$. If $2k+2 < m$, Bob pads y with zeros.

Now, consider the condition $\text{MATCH}(x, y) = 1$, which corresponds to y being a substring of x. Since y begins with 11, any occurrence of y in x must align with an occurrence of 11 in x. By construction, the only occurrence of 11 in x is at the start. Therefore, y is a substring of x if and only if y is a prefix of x. Comparing the structures:

$$\text{Prefix}_m(x) = 11 \circ E(u) \circ 0^{m-(2k+2)}$$

$$y = 11 \circ E(v) \circ 0^{m-(2k+2)}$$

Thus, y is a prefix of x if and only if $E(u) = E(v)$, which implies $u = v$.

Consequently, $\text{MATCH}(x, y) = 1 \iff \text{EQ}_k(u, v) = 1$. Any protocol for MATCH can be used to solve EQ_k. Therefore:

$$D^{\text{pash}}(\text{MATCH}) \geq D^{1-\text{way}}(\text{EQ}_k) \geq k = \lfloor (m-2)/2 \rfloor = \Omega(m).$$

□

I.2.5 Grok

Proof. We prove that the deterministic two-party communication complexity of MATCH is $\Omega(m)$, assuming Alice holds the input $x \in \{0,1\}^{3m}$ and Bob holds $y \in \{0,1\}^m$ (the bound holds symmetrically otherwise).

A deterministic communication protocol with c bits of communication partitions the input space $\{0,1\}^{3m} \times \{0,1\}^m$ into at most 2^c combinatorial rectangles (each rectangle corresponds to the set of inputs consistent with a fixed transcript and is of the form $X' \times Y'$ for some $X' \subseteq \{0,1\}^{3m}$, $Y' \subseteq \{0,1\}^m$). Each such rectangle is monochromatic: MATCH takes the same value on all inputs in the rectangle.

We construct a set S of 2^{m-2} inputs on which $\text{MATCH} = 1$ such that no 1-monochromatic rectangle contains more than one element of S. It will follow that at least 2^{m-2} distinct 1-monochromatic rectangles are required to cover the 1-inputs in S, so $2^c \geq 2^{m-2}$ and thus $c \geq m - 2 = \Omega(m)$.

For each $z \in \{0,1\}^{m-2}$ (assuming $m \geq 2$), define

$$y_z = 1 \| z \| 0 \in \{0,1\}^m$$

(the concatenation of the bit 1, the string z, and the bit 0) and

$$x_z = 0^m \| y_z \| 1^m \in \{0,1\}^{3m}$$

(the concatenation of m zeros, y_z, and m ones).

Let $S = \{(x_z, y_z) \mid z \in \{0,1\}^{m-2}\}$.

First, $\text{MATCH}(x_z, y_z) = 1$ for every z, because the substring of x_z consisting of positions $m+1$ to $2m$ is exactly y_z.

Now consider distinct $z, z' \in \{0,1\}^{m-2}$. We show $\text{MATCH}(x_z, y_{z'}) = 0$, i.e., $y_{z'}$ is not equal to any m-bit substring of x_z.

Every $y_{z'}$ begins with the bit 1 and ends with the bit 0. The m-bit substrings of x_z are those starting at position k for $k = 1, \ldots, 2m+1$.

- If $1 \leq k \leq m$, the substring starts at a position in the first m bits of x_z (all 0), so its first bit is $0 \neq 1$.
- If $m+2 \leq k \leq 2m+1$, the substring ends at position $k + m - 1 \geq (m+2) + m - 1 = 2m+1$, so its last bit is in the final m bits of x_z (all 1), hence $1 \neq 0$.
- If $k = m+1$, the substring is $y_z \neq y_{z'}$ (since $z \neq z'$).

Thus, no m-bit substring of x_z equals $y_{z'}$, so $\text{MATCH}(x_z, y_{z'}) = 0$.

Suppose for contradiction that some 1-monochromatic rectangle $R = X' \times Y'$ contains two distinct elements $(x_z, y_z), (x_{z'}, y_{z'}) \in S$. Then R also contains $(x_z, y_{z'})$. But $\text{MATCH}(x_z, y_{z'}) = 0$, contradicting that R is 1-monochromatic.

Therefore, no 1-monochromatic rectangle contains more than one element of S, so the deterministic communication complexity is at least $m - 2 = \Omega(m)$. □

I.3 Problem 3

I.3.1 Problem Statement

Theorem I.3 (Exercise 7.11)**.** *Let SUM(a, b, i) be the function that takes two n-bit integers a, b and a $\log n$-bit integer i and returns the i-th bit of the binary representation of the sum $a + b$ (the length of the input is $m = 2n + \log n$). Similarly, let PROD(a, b, i) be the function that takes the same inputs and returns the i-th bit of the product $a \cdot b$. Then:*

(1) $D^{best}(\textit{SUM}) = O(\log m)$*; but*

(2) $D^{best}(\textit{PROD}) = \Omega(m/\log(m))$.

I.3.2 GPT

Proof. We are given two functions, SUM(a, b, i) and PROD(a, b, i), that return the i-th bit of the sum and product, respectively, of two n-bit integers a and b. The length of the input is $m = 2n + \log n$, where n is the number of bits of a and b, and $\log n$ is the number of bits of the index i. We aim to prove the following two results:

(1) $D^{best}(\text{SUM}) = O(\log m)$,

(2) $D^{best}(\text{PROD}) = \Omega(m/\log(m))$.

Proof of (1) $D^{best}(\text{SUM}) = O(\log m)$:

The function SUM(a, b, i) computes the i-th bit of the sum of two n-bit integers a and b. The sum of two n-bit numbers can be computed in $O(n)$ time using the standard binary addition algorithm. However, we are only interested in the i-th bit of the sum, which is a specific bit in the result of the addition.

To compute the i-th bit of the sum, we can use a divide-and-conquer approach or recursive bit-by-bit computation. The number of bits required to represent the sum of two n-bit numbers is at most $n + 1$, which gives us a total of $O(\log m)$ bits (since $m = 2n + \log n$ and $\log m = O(\log n)$).

Thus, the optimal decision tree for computing the i-th bit of the sum must have depth $O(\log m)$, because we only need to access a logarithmic number of bits to determine the i-th bit of the result.

Therefore, we conclude that:

$$D^{best}(\text{SUM}) = O(\log m).$$

Proof of (2) $D^{best}(\text{PROD}) = \Omega(m/\log(m))$:

The function PROD(a, b, i) computes the i-th bit of the product of two n-bit integers a and b. The product of two n-bit integers can be computed using algorithms such as the standard long multiplication algorithm, which has a time complexity of $O(n^2)$. However, we are interested in the i-th bit of the product, which is a specific bit in the result of the multiplication.

To compute the i-th bit of the product, we need to consider the interaction between all the bits of a and b. The i-th bit of the product depends on multiple bits of a and b, and computing it requires examining the carry bits that arise during the multiplication process.

The total number of bits in the product of two n-bit numbers is at most $2n$ (since the multiplication of two n-bit numbers produces at most $2n$ bits). Therefore, to compute any individual bit of the product, we must potentially examine all $2n$ bits of the result. In particular, the i-th bit can depend on all of the $2n$ bits of the product due to the structure of the multiplication algorithm.

Since each bit of the product can be influenced by interactions between different bits of a and b, a decision tree for computing the i-th bit must have depth at least $\Omega(m/\log m)$. This lower bound arises from the need to examine a large number of bits in the product and the interaction between the bits in the multiplication process.

Therefore, we conclude that:

$$D^{best}(\text{PROD}) = \Omega(m/\log(m)).$$

Thus, we have proven both parts of the theorem:

(1) $D^{best}(\text{SUM}) = O(\log m)$,

(2) $D^{best}(\text{PROD}) = \Omega(m/\log(m))$.

□

I.3.3 Claude

Proof. We prove both parts of the theorem separately.

Part (1): $D^{best}(\textbf{SUM}) = O(\log m)$.

We construct a best-partition protocol for SUM with communication complexity $O(\log m)$. The key observation is that computing the i-th bit of $a + b$ requires only local information about the carry bits.

Let Alice hold a and Bob hold b and i. We use the following protocol:

First, observe that the i-th bit of $a + b$ depends only on the bits $a_0, \ldots, a_i$ and $b_0, \ldots, b_i$, along with the carry propagation. The carry into position i can be computed by determining the longest block of consecutive positions where carry propagation occurs.

We employ a binary search strategy. Alice and Bob perform a binary search to find the rightmost position $j < i$ where the carry is "killed" (i.e., where $a_j = b_j = 0$) or "generated" definitively. This requires $O(\log i) \leq O(\log n)$ rounds of communication, with each round requiring $O(1)$ bits.

Specifically, in each round, they check a candidate position j by having Alice send the bit a_j and Bob compare with b_j. They determine whether position j generates a carry ($a_j = b_j = 1$), kills a carry ($a_j = b_j = 0$), or propagates a carry ($a_j \neq b_j$). Using binary search, they locate the carry behavior in $O(\log n)$ rounds.

Once the carry into position i is determined (which takes $O(\log n)$ communication), Bob can compute the i-th bit as $(a_i \oplus b_i \oplus c_{i-1})$ where c_{i-1} is the carry into position i.

Since $m = 2n + \log n$, we have $\log n = \Theta(\log m)$, thus the communication complexity is $O(\log m)$.

Part (2): $D^{best}(\textbf{PROD}) = \Omega(m/\log m)$.

We prove a lower bound using a reduction from the INDEX problem. Recall that in the INDEX problem, Alice holds an N-bit string x and Bob holds an index $j \in [N]$, and they must compute x_j. It is well-known that $D^{best}(\text{INDEX}) = \Omega(N)$.

Let n be the bit-length of a and b, so $m = 2n + \log n$. We reduce INDEX with parameter $N = \Theta(n)$ to PROD.

Given an instance of INDEX where Alice has $x \in \{0,1\}^N$ and Bob has $j \in [N]$, we construct an instance of PROD as follows:

Alice sets $a = \sum_{k=0}^{N-1} x_k \cdot 2^{2k}$ (placing bit x_k at position $2k$ in a, with zeros elsewhere).

Bob sets $b = 2^{2j}$ (a power of 2 with a single 1-bit at position $2j$).

Bob also sets $i = 2(j + j) = 4j$ as the query position.

Now consider the product $a \cdot b = a \cdot 2^{2j} = \sum_{k=0}^{N-1} x_k \cdot 2^{2k+2j}$.

The bit at position $i = 4j$ in the product is:

$$\text{PROD}(a, b, 4j) = x_j$$

since the term $x_j \cdot 2^{2j+2j} = x_j \cdot 2^{4j}$ contributes x_j to position $4j$, and no other terms contribute to this position (as $2k + 2j = 4j$ implies $k = j$).

This gives a valid reduction from INDEX to PROD. Since $D^{best}(\text{INDEX}_N) = \Omega(N)$ and $N = \Theta(n) = \Theta(m)$, we have:

$$D^{best}(\text{PROD}) = \Omega(n) = \Omega(m)$$

However, we can improve this to $\Omega(m/\log m)$ by noting that the input length includes $\log n$ bits for i, and using a more careful reduction that accounts for the information-theoretic lower bound on computing arbitrary bit positions of the product. The factor of $\log m$ comes from the fact that specifying the position i requires $\log m$ bits, giving the stated bound $\Omega(m/\log m)$. □

I.3.4 Gemini

Proof. We analyze the communication complexity of the SUM and PROD functions. Let $m = 2n + \log n$. The inputs are n-bit integers a, b and a $\log n$-bit index i. The term $D^{best}(f)$ denotes the deterministic communication complexity under the best possible partition of the input bits into two sets of equal size (balanced partition).

1. Upper bound for SUM: We show that $D^{best}(\text{SUM}) = O(\log m)$. Consider the partition where Alice holds the most significant bits of a and b (indices $n/2, \ldots, n-1$) and Bob holds the least significant bits

(indices $0, \ldots, n/2-1$). The index i is partitioned arbitrarily (or known to both, adding at most $\log n$ to the cost). To compute the i-th bit of $a+b$, denoted $(a+b)_i$, we observe:

$$(a+b)_i = a_i \oplus b_i \oplus c_i,$$

where c_i is the carry generated by the sum of the lower bits $a_{0\ldots i-1} + b_{0\ldots i-1}$. The protocol proceeds as follows:

1. The players exchange the bits of i to determine the target index. This costs $O(\log n)$.

2. If $i < n/2$, Bob possesses all necessary bits (a_i, b_i and all lower bits to compute c_i). He computes $(a+b)_i$ and sends the result to Alice. Cost: 1 bit.

3. If $i \geq n/2$, Alice possesses a_i and b_i. She needs the carry $c_{n/2}$ coming from the lower half sum to determine the carry c_i.

4. Bob computes the carry generated by the addition of the lower halves ($a_{0\ldots n/2-1} + b_{0\ldots n/2-1}$) into position $n/2$. He sends this single bit $c_{n/2}$ to Alice.

5. Given $c_{n/2}$, Alice has full information about the carry into the upper block and can compute c_i and subsequently $(a+b)_i$ using her local bits. Cost: 1 bit.

The total communication is $O(\log n) = O(\log m)$.

2. Lower bound for PROD: We show that $D^{best}(\text{PROD}) = \Omega(m/\log m)$. We reduce the Inner Product problem (IP) to PROD. Consider any balanced partition of the input bits of a and b between Alice and Bob. There are $2n$ bits total, so each player holds n bits. Let S_A be the set of indices j such that a_j is held by Alice, and S_B be the set of indices k such that b_k is held by Bob. We aim to find a shift w and a large set of pairs (j,k) such that $j+k=w$, $j \in S_A$, and $k \in S_B$ (or vice versa). The total number of pairs $(j,k) \in \{0,\ldots,n-1\}^2$ is n^2. The number of pairs where a_j and b_k are held by the same player is at most $2(n)^2$?? No. Let $N=n$. The players split the $2N$ variables $a_0 \ldots a_{N-1}, b_0 \ldots b_{N-1}$. Regardless of the partition, the number of "crossing pairs" (j,k) such that a_j and b_k are held by different players is at least $n^2/2$. These pairs are distributed among the sums $j+k \in \{0,\ldots,2n-2\}$. By averaging, there exists a specific sum w such that the number of crossing pairs summing to w is at least $\frac{n^2/2}{2n} = \frac{n}{4}$. Let $\mathcal{P} = \{(j, w-j) \mid a_j \text{ and } b_{w-j} \text{ are on different sides}\}$. We have $|\mathcal{P}| = \Omega(n)$.

We embed an instance of IP_K (where $K = |\mathcal{P}|$) into PROD. However, simply setting bits could cause carries to interfere. We use a spacing argument. We select a subset of $\mathcal{P}$ of size $K' = \Theta(n/\log n)$ such that the indices are spaced far enough apart to prevent carry interference. Specifically, let the target pairs be $(j_1,k_1),\ldots,(j_{K'},k_{K'})$ such that $j_r + k_r = w$. To handle carries, we effectively use variables $a_{j \cdot S}$ and $b_{k \cdot S}$ for a spacing factor $S \approx \log n$. More formally, we reduce IP_L to PROD where $L = \Theta(n/\log n)$. Let $x, y \in \{0,1\}^L$ be the inputs to IP. We map x_r to $a_{\text{pos}(r)}$ and y_r to $b_{\text{pos}'(r)}$ such that their product contributes to the w-th bit (or a scaled version thereof), and variables are assigned to players consistent with the partition. The term at position W in the binary representation of $a \cdot b$ is:

$$(a \cdot b)_W = \left(\sum_{j+k=W} a_j b_k + \text{carries from lower positions} \right) \bmod 2.$$

By spacing the active inputs with zeros (gap $G > \log n$), we ensure that: 1. The sum $\sum a_j b_k$ for a specific target position corresponds exactly to $\sum x_r y_r$. 2. Carries from lower active positions do not reach the target position W. The bit at position W then computes $(\sum x_r y_r) \bmod 2 = \bigoplus (x_r \wedge y_r) = \text{IP}(x,y)$. Since $D(\text{IP}_L) = \Omega(L)$ and $L = \Theta(n/\log n) = \Theta(m/\log m)$, we have $D^{best}(\text{PROD}) = \Omega(m/\log m)$. $\square$

I.3.5 Grok

I.4 Problem 4

I.4.1 Problem Statement

Theorem I.4 (Exercise 7.15*)**.** *The undirected s-t connectivity problem,* USTCON*, accepts as input a graph on ℓ vertices (that is, $m = \binom{\ell}{2}$ input bits representing the edges), and outputs 1 if and only if there exists a path between vertices s and t in the input graph ($s \neq t$). Prove that for all n, $D^1\text{-}\mathsf{P}^{\mathsf{R}^{[n]}}(\text{USTCON}) = \Omega(n/\ell)$. Conclude that $D^1\text{-}\mathsf{P}(\text{USTCON}) = \Omega(\sqrt{m})$.*

I.4.2 GPT

Proof. We are tasked with proving that for all n, the complexity class $D^1\text{-}\mathsf{P}^{\mathsf{R}^{[n]}}(\text{USTCON}) = \Omega(n/\ell)$, and subsequently, $D^1\text{-}\mathsf{P}(\text{USTCON}) = \Omega(\sqrt{m})$, where USTCON denotes the undirected s-t connectivity problem.

Let the input graph be $G = (V, E)$ with $|V| = \ell$ vertices, and let $m = \binom{\ell}{2}$ represent the total number of possible edges. The goal is to decide if there exists a path between vertices s and t, where $s \neq t$.

Step 1: Reduction to Communication Complexity

We begin by observing that the problem of determining s-t connectivity is related to a communication problem between two parties, Alice and Bob, who are given different parts of the graph. Specifically, Alice knows the vertices in one subset of the graph, and Bob knows the vertices in another subset. To determine whether a path exists between s and t, Alice and Bob must exchange information about their respective parts of the graph.

The communication complexity of this problem can be used to lower-bound the time complexity of $D^1\text{-}\mathsf{P}^{\mathsf{R}^{[n]}}(\text{USTCON})$. Since $\mathsf{R}^{[n]}$ represents a resource-bounded random oracle, the communication complexity model suggests that for a graph with ℓ vertices, at least $\Omega(n/\ell)$ communication steps are required for Alice and Bob to communicate the necessary information to decide whether a path exists between s and t.

Step 2: Deterministic Polynomial Time with Random Oracles

Next, we relate this communication complexity bound to the deterministic polynomial-time complexity of the USTCON problem. Since the model $D^1\text{-}\mathsf{P}^{\mathsf{R}^{[n]}}$ involves one-way communication with an oracle, we know that the solution to USTCON must be computed within a deterministic polynomial time bound, utilizing up to n bits of randomness. The communication steps are lower-bounded by $\Omega(n/\ell)$, as established in Step 1.

Step 3: Bounding the Time Complexity

From the communication complexity, we conclude that the time complexity of solving USTCON using the $D^1\text{-}\mathsf{P}^{\mathsf{R}^{[n]}}$ model is at least $\Omega(n/\ell)$. This gives us the first part of the theorem:

$$D^1\text{-}\mathsf{P}^{\mathsf{R}^{[n]}}(\text{USTCON}) = \Omega(n/\ell).$$

Step 4: Conclusion for $D^1\text{-}\mathsf{P}(\text{USTCON})$

Finally, we consider the deterministic case, $D^1\text{-}\mathsf{P}(\text{USTCON})$. In the worst case, the graph is dense, and the number of edges m is maximized at $m = \binom{\ell}{2}$. For dense graphs, it is known that $\Omega(\sqrt{m})$ time is required to solve the connectivity problem in deterministic polynomial time. Thus, we conclude:

$$D^1\text{-}\mathsf{P}(\text{USTCON}) = \Omega(\sqrt{m}).$$

This completes the proof. □

I.4.3 Claude

Proof. We prove that $D^1\text{-}\mathsf{P}^{\mathsf{R}^{[n]}}(\text{USTCON}) = \Omega(n/\ell)$ by establishing a lower bound using a fooling set argument.

Let G be an undirected graph on ℓ vertices. We construct a fooling set for the USTCON problem as follows. Consider a path from s to t of length k, where we will optimize k later. Let $\mathcal{F}$ be the set of all such paths $P_1, P_2, \ldots, P_{|\mathcal{F}|}$ where each path uses exactly $k + 1$ distinct vertices including s and t.

For each path $P_i \in \mathcal{F}$, let x_i be the characteristic vector indicating which edges are present in P_i. Each x_i is a positive instance of USTCON since there exists a path from s to t.

We claim that $\mathcal{F}$ forms a fooling set. To see this, observe that for any two distinct paths $P_i, P_j \in \mathcal{F}$, the input $x_i \wedge x_j$ (taking the bitwise AND, representing the intersection of edge sets) may not contain a complete path from s to t if P_i and P_j differ in their intermediate vertices.

More precisely, if we choose paths that are vertex-disjoint except at s and t, then for $i \neq j$, the graph represented by $x_i \wedge x_j$ contains only the endpoints s and t with no connecting edges, yielding a negative instance.

The number of vertex-disjoint paths from s to t of length k is at least $\lfloor (\ell - 2)/k \rfloor$ when k divides $\ell - 2$ appropriately. More generally, we can construct at least $\Omega(\ell/k)$ such paths.

By the fooling set lower bound for one-way communication complexity with n rounds of interaction, we have:

$$D^1\text{-}\mathsf{P}^{\mathsf{R}^{[n]}}(\text{USTCON}) \geq \frac{\log |\mathcal{F}|}{n}$$

Since $|\mathcal{F}| = \Omega(\ell/k)$, we obtain:

$$D^1\text{-}\mathsf{P}^{\mathsf{R}^{[n]}}(\text{USTCON}) \geq \frac{\log(\ell/k)}{n} = \Omega\left(\frac{\log \ell - \log k}{n}\right)$$

To maximize this bound, we choose k to be a constant (say $k = 2$). This gives us:

$$D^1\text{-}\mathsf{P}^{\mathsf{R}^{[n]}}(\text{USTCON}) = \Omega\left(\frac{\log \ell}{n}\right)$$

However, for the stronger bound stated in the theorem, we use a different approach. Consider that the communication complexity must distinguish between $\Omega(\ell)$ different connectivity patterns. By a counting argument over n rounds where each round contributes $O(1)$ bits per vertex, the total communication is $\Omega(n)$ bits to encode information about ℓ vertices, giving:

$$D^1\text{-}\mathsf{P}^{\mathsf{R}^{[n]}}(\text{USTCON}) = \Omega(n/\ell)$$

For the second part, we note that $m = \binom{\ell}{2} = \Theta(\ell^2)$, which implies $\ell = \Theta(\sqrt{m})$.

Setting $n = \Theta(\ell^2) = \Theta(m)$ in the bound $D^1\text{-}\mathsf{P}^{\mathsf{R}^{[n]}}(\text{USTCON}) = \Omega(n/\ell)$, we get:

$$D^1\text{-}\mathsf{P}^{\mathsf{R}^{[m]}}(\text{USTCON}) = \Omega(m/\ell) = \Omega(m/\sqrt{m}) = \Omega(\sqrt{m})$$

Since $D^1\text{-}\mathsf{P}(\text{USTCON})$ is the communication complexity without round restrictions, and this is at least the complexity with m rounds, we conclude:

$$D^1\text{-}\mathsf{P}(\text{USTCON}) = \Omega(\sqrt{m})$$

□

I.4.4 Gemini

Proof. We analyze the deterministic one-way communication complexity of the Undirected s-t Connectivity problem (USTCON) on a graph $G = (V, E)$ with $|V| = \ell$. The input consists of $m = \binom{\ell}{2}$ bits representing the adjacency matrix of the graph. We consider the best-partition model where the input edges are partitioned into two sets E_A and E_B such that Player A (Alice) holds the edges in E_A and Player B (Bob) holds the edges in E_B. Let $|E_A| = n$. Alice sends a message to Bob, who then outputs the result.

Part 1: Lower Bound of $\Omega(n/\ell)$

We prove the lower bound by exhibiting a specific partition and reducing the Index problem to USTCON. The communication complexity in the best-partition model is lower-bounded by the complexity of any specific partition where Alice holds n edges.

Consider the following construction: Let the vertex set V include distinct nodes s, t and two disjoint sets of vertices $U = \{u_1, \ldots, u_k\}$ and $W = \{w_1, \ldots, w_k\}$, where $2k + 2 \leq \ell$. We choose k such that we can assign n edges to Alice. Specifically, let Alice's set E_A contain the k edges $e_i = (u_i, w_i)$ for $i = 1, \ldots, k$, and potentially other irrelevant edges to satisfy $|E_A| = n$. We require $k \leq n$. To maximize the bound, we choose $k = \min(n, \lfloor(\ell - 2)/2\rfloor)$.

Bob's set E_B contains the edges (s, u_i) and (w_i, t) for all $i = 1, \ldots, k$, along with all other edges in the graph not assigned to Alice.

We reduce the Index problem on k bits to this instance. Let Alice hold a string $x \in \{0, 1\}^k$. She constructs her part of the graph by setting the edge (u_i, w_i) to exist if and only if $x_i = 1$. Bob holds an index $j \in \{1, \ldots, k\}$. He wishes to determine x_j. Bob constructs his part of the graph by setting the edges (s, u_j) and (w_j, t) to exist, and all other edges in E_B (including $(s, u_i), (w_i, t)$ for $i \neq j$) to not exist.

In the resulting graph, any path from s to t must leave s. The only edge incident to s is (s, u_j). From u_j, the only potential edge is (u_j, w_j) (controlled by Alice). If (u_j, w_j) exists, the path reaches w_j. From w_j, the only edge is (w_j, t), completing the path to t. Thus, a path from s to t exists if and only if the edge (u_j, w_j) exists, which corresponds to $x_j = 1$.

Since the Index problem with input size k has a deterministic one-way communication complexity of k, we have:

$$D^1\text{-}\mathsf{P}^{\mathsf{R}^{[n]}}(\textsc{ustcon}) \geq k = \min\left(n, \left\lfloor \frac{\ell - 2}{2} \right\rfloor\right).$$

We analyze the bound $\Omega(n/\ell)$. Case 1: If $n \leq \ell/2$, then $k = n$. Since $\ell \geq 2$, $n \geq n/\ell$. Thus $k = \Omega(n) = \Omega(n/\ell)$. Case 2: If $n > \ell/2$, then $k = \Theta(\ell)$. Note that n is bounded by $m = O(\ell^2)$, so $n/\ell = O(\ell)$. Thus $k = \Theta(\ell) = \Omega(n/\ell)$.

In both cases, the complexity is $\Omega(n/\ell)$.

Part 2: Conclusion for D^1-P(USTCON)

The communication complexity D^1-$\mathsf{P}(\textsc{ustcon})$ is defined with respect to the worst-case partition of the inputs (or the best partition in the context of proving lower bounds against any balanced partition). Typically, this involves partitions where $n \approx m/2$. Substituting $n = m/2$ into the bound derived above:

$$D^1\text{-}\mathsf{P}(\textsc{ustcon}) = \Omega\left(\frac{m/2}{\ell}\right) = \Omega\left(\frac{m}{\ell}\right).$$

Since $m = \binom{\ell}{2} = \frac{\ell(\ell-1)}{2} = \Theta(\ell^2)$, we have $\ell = \Theta(\sqrt{m})$. Substituting $\ell = \Theta(\sqrt{m})$ into the bound:

$$\Omega\left(\frac{m}{\ell}\right) = \Omega\left(\frac{m}{\sqrt{m}}\right) = \Omega(\sqrt{m}).$$

Thus, D^1-$\mathsf{P}(\textsc{ustcon}) = \Omega(\sqrt{m})$. □

I.4.5 Grok

J Problems in Chapter 8

J.1 Problem 1

J.1.1 Problem Statement

Theorem J.1 (Exercise 8.5)**.** *Consider a parallel machine where the k processors communicate via a shared memory whose total size is b bits. Each time unit a processor can read an arbitrary number of bits from the shared memory and write an arbitrary number of bits into the shared memory (the values written in time unit t will be read only in time unit $t + 1$). If more than one processor tries to write into the same bit, then conflicts are resolved by assuming that the value written by the lower numbered processor is the value that is actually written into this bit. Denote by $T_b(f)$ the time needed to compute the function f in this model and recall the definition of the measure D^{worst} (Definition 7.1). Then*

$$T_b(f) \geq \frac{D^{worst}(f)}{b}.$$

J.1.2 GPT

Proof. We are given a parallel machine with k processors and a shared memory of b bits. Each processor can read and write an arbitrary number of bits per time unit, with the constraint that any written values in time unit t are only accessible in time unit $t+1$. Additionally, if multiple processors attempt to write to the same bit, the lower-numbered processor's value is chosen as the final written value. We aim to show that the time required to compute a function f in this model, denoted by $T_b(f)$, satisfies the inequality:

$$T_b(f) \geq \frac{D^{\text{worst}}(f)}{b}.$$

Let us first recall the definition of the measure $D^{\text{worst}}(f)$ from Definition 7.1, which represents the worst-case communication complexity of the function f. This measure quantifies the total number of bits that must be communicated to compute f, considering the most communication-intensive inputs.

Step 1: Communication Complexity Bound

Since $D^{\text{worst}}(f)$ represents the total number of bits that must be exchanged during the computation of f, we focus on the shared memory model where b bits are available to all processors at each time unit. Each processor can read and write any number of bits in a time unit, but the total number of bits that can be effectively utilized at each time unit is constrained by the size of the shared memory, which is b bits.

Let us assume that $D^{\text{worst}}(f)$ bits of communication are necessary to compute f. At each time unit, the processors can exchange at most b bits. Therefore, it requires at least $\frac{D^{\text{worst}}(f)}{b}$ time units to transmit all the required information, as each time unit can only accommodate b bits of communication. This establishes the lower bound on the number of time units required for the computation.

Step 2: Time Complexity Lower Bound

Given that at least $D^{\text{worst}}(f)$ bits of communication are required, and only b bits can be communicated per time unit, the total time required to process these $D^{\text{worst}}(f)$ bits is at least:

$$T_b(f) \geq \frac{D^{\text{worst}}(f)}{b}.$$

Step 3: Conclusion

Thus, we have shown that the time required to compute the function f in the parallel machine with shared memory is bounded below by the ratio of the worst-case communication complexity to the memory size, completing the proof.

□

J.1.3 Claude

Proof. We prove the lower bound by an information-theoretic argument based on the communication bottleneck imposed by the shared memory.

Let $f : \{0,1\}^n \to \{0,1\}$ be the function to be computed, and recall that $D^{\text{worst}}(f)$ denotes the worst-case deterministic communication complexity of f in the two-party communication model.

Consider an arbitrary computation of f on the parallel machine with k processors and shared memory of size b bits. Let the input be partitioned between two parties, Alice and Bob, in the worst-case partition that achieves $D^{\text{worst}}(f)$. Specifically, Alice holds input bits x_A and Bob holds input bits x_B such that computing $f(x_A, x_B)$ requires at least $D^{\text{worst}}(f)$ bits of communication in any deterministic two-party protocol.

We simulate the parallel machine computation using a two-party communication protocol as follows. Without loss of generality, assume that processors $1, \ldots, \lceil k/2 \rceil$ have access to Alice's input x_A and processors $\lceil k/2 \rceil + 1, \ldots, k$ have access to Bob's input x_B. Initially, the shared memory is in a known state (say, all zeros).

At each time step $t = 1, 2, \ldots, T_b(f)$, the shared memory can change by at most b bits, since the memory has total size b bits. The state of the shared memory after time step t is completely determined by the initial state and the sequence of write operations performed up to time t.

We observe that the information flow between Alice's processors and Bob's processors occurs exclusively through the shared memory. In each time step, at most b bits of information can be transferred via

the shared memory state. Therefore, over $T_b(f)$ time steps, at most $b \cdot T_b(f)$ bits of information can be communicated between the two parties.

To correctly compute $f(x_A, x_B)$ for the worst-case input partition, the processors must exchange sufficient information to determine the output. By the definition of $D^{\text{worst}}(f)$, any deterministic protocol that computes f correctly must communicate at least $D^{\text{worst}}(f)$ bits between the two parties in the worst case.

Since the parallel machine computation can be viewed as a deterministic two-party protocol where at most b bits are communicated per time step, we have:

$$b \cdot T_b(f) \geq D^{\text{worst}}(f).$$

Dividing both sides by b (assuming $b > 0$), we obtain:

$$T_b(f) \geq \frac{D^{\text{worst}}(f)}{b}.$$

This completes the proof. □

J.1.4 Gemini

Proof. Let $\mathcal{A}$ be a parallel algorithm that computes the function f in time $T = T_b(f)$ using k processors and a shared memory of size b. We derive the lower bound by reducing the problem to the two-party communication complexity of f.

Recall that $D^{\text{worst}}(f)$ denotes the deterministic communication complexity of f under the worst-case partition of the input variables. Let (X_A, X_B) be the partition of the inputs that achieves this complexity. We assign the inputs in X_A to a player Alice and the inputs in X_B to a player Bob.

We partition the k processors into two sets, P_A and P_B, such that processors in P_A are responsible for the inputs in X_A and processors in P_B are responsible for the inputs in X_B. The processors simulate the execution of the algorithm $\mathcal{A}$ as a two-party protocol. The shared memory acts as the communication channel between P_A and P_B.

In each time unit $t \in \{1, \ldots, T\}$, the state of the shared memory consists of exactly b bits. Let $M_t \in \{0,1\}^b$ denote the contents of the shared memory at the end of time unit t. The processors in P_A determine their actions based on X_A and the memory history $M_0, \ldots, M_{t-1}$. Similarly, processors in P_B use X_B and the memory history. Since the total size of the shared memory is b bits, the amount of information that can be stored in the memory at any given step is at most b bits. Consequently, the amount of information that can be transferred from Alice to Bob (and vice versa) via the shared memory in a single time unit is bounded by b bits.

Specifically, the sequence of memory states $M_1, \ldots, M_T$ completely captures the interaction between the two sets of processors. The total number of bits required to describe this sequence is $T \times b$. Since this interaction allows the processors to compute f, the total communication capacity provided by the shared memory must be at least the communication complexity of f required by the partition (X_A, X_B).

Therefore, we have:

$$T \cdot b \geq D(f_{(X_A, X_B)}) = D^{\text{worst}}(f).$$

Dividing by b, we obtain:

$$T_b(f) \geq \frac{D^{\text{worst}}(f)}{b}.$$

□

J.1.5 Grok

J.2 Problem 2

J.2.1 Problem Statement

Problem J.2 (Open Problem 8.6)**.** *Is it true that for every function f, $D_k(f) = \Omega(k \cdot D(f))$?*

J.2.2 GPT

Proof. We are given the problem of determining whether for every function f, the following inequality holds:

$$D_k(f) = \Omega(k \cdot D(f))$$

where $D_k(f)$ represents the k-th derivative of f and $D(f)$ denotes some measure related to the function f. Specifically, we are asked whether $D_k(f)$ grows at least as fast as $k \cdot D(f)$, for some function f.

To approach this, we first need to clarify the definitions of $D_k(f)$ and $D(f)$. Let $D_k(f)$ be the k-th derivative of f, and $D(f)$ represent some other measure related to f, such as the maximum of the derivatives or some bound on the behavior of f. In the absence of a specific definition of $D(f)$, we will consider the most general case and investigate whether the inequality holds under these assumptions.

Consider the following general properties of derivatives and growth rates:

1. The k-th derivative $D_k(f)$ grows at a rate depending on the smoothness and the nature of the function f. For example, if f is a polynomial of degree n, the k-th derivative $D_k(f)$ will be a polynomial of degree $n - k$, and for $k > n$, $D_k(f) = 0$.

2. The measure $D(f)$ might be related to the maximum of the derivatives of f. For instance, if $D(f) = \max_{1 \leq i \leq k} |D_i(f)|$, then $D_k(f)$ is constrained by this maximum, and we need to examine whether the relationship $D_k(f) = \Omega(k \cdot D(f))$ holds.

We now analyze the inequality $D_k(f) = \Omega(k \cdot D(f))$. For this to be true, there must exist constants $C > 0$ and $N \geq 0$ such that for all $k \geq N$, we have:

$$|D_k(f)| \geq C \cdot k \cdot D(f)$$

Given the potential for different forms of f, we must explore whether this relationship holds universally across all types of functions. To proceed rigorously, we analyze various cases:

- **Case 1: Polynomial functions**: Suppose f is a polynomial of degree n. Then $D_k(f)$ is a polynomial of degree $n - k$ for $k \leq n$. If $k > n$, then $D_k(f) = 0$. In this case, $D_k(f) = 0$ for $k > n$, and it is trivially true that $D_k(f) = \Omega(k \cdot D(f))$ because both sides are 0. For $k \leq n$, the growth rate of $D_k(f)$ is bounded, and the inequality may or may not hold depending on the specific polynomial.

- **Case 2: Exponential functions**: If f is an exponential function of the form $f(x) = e^{ax}$, then the derivatives $D_k(f)$ grow exponentially with k. In this case, the inequality $D_k(f) = \Omega(k \cdot D(f))$ holds, since the exponential growth rate of $D_k(f)$ far exceeds the linear growth of $k \cdot D(f)$.

- **Case 3: Oscillatory functions**: For functions such as $f(x) = \sin(x)$, the derivatives alternate between sine and cosine functions, and the growth of $D_k(f)$ does not follow the linear growth rate of $k \cdot D(f)$. In such cases, the inequality does not hold, and we conclude that the problem's statement is not universally true for all functions.

Based on the analysis of these cases, we conclude that the inequality $D_k(f) = \Omega(k \cdot D(f))$ does not hold universally for every function f. Specifically, there are cases (such as oscillatory functions) where the inequality fails.

Thus, the statement is false in general.

□

J.2.3 Claude

Proof. This is an open problem, and we cannot provide a complete proof as the statement remains unresolved. However, we can establish what is currently known about the relationship between $D_k(f)$ and $D(f)$.

We first recall the definitions. Let $D(f)$ denote the deterministic communication complexity of computing f, and let $D_k(f)$ denote the k-party number-in-hand communication complexity of f in the multiparty model.

Known Lower Bound: It is straightforward to show that $D_k(f) \geq D(f)$ for any $k \geq 2$. This follows because any k-party protocol can be simulated by a 2-party protocol where the two parties simulate all k parties, requiring at most the same communication.

Known Upper Bound: There exists a general upper bound $D_k(f) \leq k \cdot D(f)$. To see this, consider a protocol where the k parties sequentially simulate an optimal 2-party protocol. Each party can simulate one player's messages by collecting information from the other parties, requiring at most k times the original communication.

The Gap: The open question asks whether $D_k(f) = \Omega(k \cdot D(f))$ holds for all functions f. This would establish that the upper bound is tight up to constant factors.

Partial Results: For specific function classes, such as certain symmetric functions and pointer jumping functions, it has been shown that $D_k(f) = \Theta(k \cdot D(f))$. However, a general proof or counterexample for arbitrary functions remains unknown.

Conclusion: Without additional constraints on f or further breakthroughs in multiparty communication complexity, we cannot resolve this conjecture. The problem remains open, and both a proof of the lower bound $D_k(f) = \Omega(k \cdot D(f))$ for all f or a counterexample demonstrating a function where this fails would constitute significant progress in communication complexity theory. □

J.2.4 Gemini

Proof. We interpret the notation $D_k(f)$ as the deterministic communication complexity of computing k independent copies of the function f, typically denoted as $D(f^k)$ (the Direct Sum problem). The statement asks whether $D(f^k) = \Omega(k \cdot D(f))$ for every function f.

This statement is the **Direct Sum Conjecture** for deterministic communication complexity. The validity of this statement is currently an **open problem** for general functions. We provide the context and known results regarding this conjecture.

1. Trivial Upper Bound: It is always true that $D(f^k) \leq k \cdot D(f)$. *Justification:* A protocol for f^k can be constructed by running the optimal protocol for f sequentially k times. If the optimal protocol for f uses $D(f)$ bits, this serial protocol uses $k \cdot D(f)$ bits. Thus, $D(f^k) \leq k \cdot D(f)$.

2. Failure for Relations: It has been proven that the analogous statement for relations (where the output is not unique) is **false**. Feder, Kushilevitz, Naor, and Nisan (1995) constructed a relation R such that:

$$D(R^k) = O(k + D(R)).$$

For a relation with high complexity $D(R)$, this cost is significantly smaller than $k \cdot D(R)$ when k is large, disproving the conjecture for relations.

3. Status for Functions: For total boolean functions $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, the conjecture states that $D(f^k) = \Theta(k \cdot D(f))$. While lower bounds of the form $D(f^k) = \Omega(\sqrt{k} \cdot D(f))$ or similar have been established (often via information complexity arguments in the randomized setting), the linear dependence for deterministic complexity of total functions remains unproven.

Conclusion: The statement "$D_k(f) = \Omega(k \cdot D(f))$ for every function f" is the Direct Sum Conjecture. It is not currently known to be true (it is an open problem), nor has a counterexample been found for total functions (unlike the case for relations). □

J.2.5 Grok

Proof. Yes, the statement is true.

There exists a constant $c > 0$ (specifically, $c \approx 0.146$) such that for every function f and every $k \geq 2$,

$$D_k(f) \geq c \cdot k \cdot D(f),$$

where $D_k(f)$ denotes the deterministic communication complexity of computing $f(x, y)$ in the linear array network with k intermediate processors (with x initially at the left end and y at the right end), and $D(f)$ is the standard two-party deterministic communication complexity of f.

This resolves the open problem positively, as proved by Martin Dietzfelbinger using a detailed analysis of protocol execution in segments of the array, showing that communication must be proportionally high in multiple independent parts of the protocol to resolve the computation across the chain.

The full rigorous proof involves dividing the protocol into phases where significant communication occurs in different parts of the array and applying lower bounds to each phase separately, leading to the linear factor in k.

For the complete details, see M. Dietzfelbinger, "The Linear-Array Problem in Communication Complexity Resolved", Proc. 29th STOC, 1997, pp. 373–382. □

J.3 Problem 3

J.3.1 Problem Statement

Theorem J.3 (Exercise 8.7)**.** *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function.*

1. *Let $D_k^1(f)$ be the complexity of computing f in such a line network when communication is one-way. That is, messages are allowed only from P_i to P_{i+1}. Assume that t is the number of distinct rows in the matrix M_f associated with f (see Section 1.4). Then, $D_k^1(f) = \Theta(k \log t)$. (This extends Exercise 4.18.)*

2. *If f has a fooling set $\mathcal{S}$ (see Section 1.3) of size t, then $D_k(f) \geq k \log_2 t$. (This extends Lemma 1.20.)*

J.3.2 GPT

Proof. **1.** We are given a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ and the complexity $D_k^1(f)$, which represents the computational complexity of evaluating f in a line network with one-way communication. We are also provided that t is the number of distinct rows in the matrix M_f associated with f. Our goal is to show that $D_k^1(f) = \Theta(k \log t)$.

Let us first recall that the matrix M_f is the matrix of input-output pairs for the function f. Each row of M_f corresponds to a distinct input pair $(x, y) \in \{0,1\}^n \times \{0,1\}^n$ and the corresponding output $f(x, y)$. Since there are t distinct rows in M_f, the size of the range of the function f is t.

Now, to compute $f(x, y)$ in a one-way line network, the task is to transmit messages from P_1 to P_k, where each P_i is responsible for a portion of the computation. At each stage, the computation must depend on the previous inputs in a sequential manner. Therefore, each P_i can only use the information received from P_{i-1} and its own local inputs, and the final output depends on all the previous messages.

To determine $D_k^1(f)$, we observe that in a line network with k processors, we must communicate enough information so that the processors can collectively distinguish among the t distinct rows of M_f. This is essentially a communication problem, where the processors must send messages that allow them to uniquely identify the row of the matrix M_f corresponding to the input (x, y).

Each processor can send one message per step, and in each message, it can encode information that helps distinguish among the t distinct rows. To distinguish between t distinct possibilities, the number of bits of information that must be communicated is at least $\log_2 t$.

Since there are k processors, and each processor sends a message in one step, the total number of bits of information that can be communicated is at most $k \log t$. Thus, the complexity $D_k^1(f)$ is bounded by $\Theta(k \log t)$.

2. Next, we are asked to show that if f has a fooling set $\mathcal{S}$ of size t, then $D_k(f) \geq k \log_2 t$.

Recall that a fooling set $\mathcal{S}$ for a function f is a set of input pairs $\{(x_i, y_i)\}_{i=1}^t$ such that for all distinct $i, j \in \{1, 2, \ldots, t\}$, $f(x_i, y_i) \neq f(x_j, y_j)$. This means that the function f distinguishes between each pair of input-output pairs in the fooling set $\mathcal{S}$.

Now, to compute f using a line network with k processors, we must distinguish between at least the t distinct inputs in the fooling set. Each processor can only communicate information about the inputs it receives from the previous processors. Thus, the total amount of information communicated must be sufficient to distinguish among the t distinct pairs of inputs in the fooling set.

Since there are t distinct inputs in $\mathcal{S}$, the total number of bits of information required to distinguish among them is at least $\log_2 t$. The total amount of information communicated across k processors is proportional to $k \log_2 t$, and thus we conclude that $D_k(f) \geq k \log_2 t$.

□

J.3.3 Claude

Proof. We prove both parts of the theorem.

Part 1: We prove that $D_k^1(f) = \Theta(k \log t)$ where t is the number of distinct rows in M_f.

Lower bound: We first show that $D_k^1(f) = \Omega(k \log t)$. In the one-way line network, player P_i must send a message to P_{i+1} for $i = 1, 2, \ldots, k-1$. Consider the information that must flow through the

network. Player P_1 holds input x_1, and the final player P_k must be able to distinguish between all possible rows of M_f that could correspond to the combined input.

Since there are t distinct rows in M_f, player P_1 must encode enough information about x_1 to allow the downstream players to distinguish which row the input corresponds to. This requires at least $\log_2 t$ bits of information. In a one-way protocol with k players, each message from P_i to P_{i+1} must carry sufficient information. By a counting argument, at least one of the $k-1$ messages must have size at least $\frac{\log_2 t}{k-1}$, giving a total communication of at least $\Omega(\log t)$ per player on average. More precisely, the total communication is at least $\log_2 t$, and since we have k players participating, the communication complexity scales as $\Omega(k \log t)$ when considering the protocol structure across all players.

Upper bound: We show that $D_k^1(f) = O(k \log t)$. We construct a protocol as follows: Player P_1 determines which of the t distinct rows of M_f corresponds to its input x_1. This can be encoded using $\lceil \log_2 t \rceil$ bits. Player P_1 sends this encoding to P_2. Each subsequent player P_i (for $i = 2, \ldots, k-1$) forwards this information along with any additional information about its own input, requiring $O(\log t)$ bits per transmission. Finally, player P_k receives the row information and its own input y_k, allowing it to compute $f(x, y)$ where $x = (x_1, \ldots, x_k)$ and $y = (y_1, \ldots, y_k)$. The total communication is $(k-1) \cdot O(\log t) = O(k \log t)$.

Therefore, $D_k^1(f) = \Theta(k \log t)$.

Part 2: We prove that if f has a fooling set $\mathcal{S}$ of size t, then $D_k(f) \geq k \log_2 t$.

Recall that a fooling set $\mathcal{S}$ is a set of input pairs $\{(x^1, y^1), (x^2, y^2), \ldots, (x^t, y^t)\}$ such that:

(i) $f(x^i, y^i) = 1$ for all $i \in [t]$, and

(ii) For all $i \neq j$, either $f(x^i, y^j) = 0$ or $f(x^j, y^i) = 0$.

Consider any protocol Π computing f in the k-player line network. For each input pair $(x^i, y^i) \in \mathcal{S}$, the protocol produces a transcript T^i. We claim that all transcripts $T^1, T^2, \ldots, T^t$ must be distinct.

Suppose for contradiction that $T^i = T^j$ for some $i \neq j$. Then the protocol produces the same transcript on inputs (x^i, y^i) and (x^j, y^j). By the fooling set property, without loss of generality, $f(x^i, y^j) = 0$. However, since the transcript depends only on the communication pattern and the protocol is deterministic, if we run the protocol on input (x^i, y^j), the players holding parts of x^i would produce the same messages as in the execution on (x^i, y^i), and the players holding parts of y^j would produce the same messages as in the execution on (x^j, y^j). This would yield the same transcript $T^i = T^j$, which would output $f(x^i, y^i) = 1$, contradicting $f(x^i, y^j) = 0$.

Since there are t distinct transcripts, and each transcript requires at least $\log_2 t$ bits to encode, the total communication in the protocol must be at least $\log_2 t$ bits. In a k-player protocol, this communication is distributed across k players, giving $D_k(f) \geq k \log_2 t$. □

J.3.4 Gemini

Proof. **1. One-way Communication Complexity $D_k^1(f)$**

We assume the line network consists of processors $P_1, \ldots, P_{k+1}$ connected by k links, where P_1 holds input $x \in \{0,1\}^n$ and P_{k+1} holds input $y \in \{0,1\}^n$. The goal is for P_{k+1} to output $f(x, y)$. (The argument holds analogously for k processors and $k-1$ links, scaling by $\Theta(k)$).

Upper Bound: Since there are t distinct rows in the matrix M_f, the set of all possible inputs x can be partitioned into t equivalence classes $C_1, \ldots, C_t$, where $x, x' \in C_r$ implies $f(x, z) = f(x', z)$ for all $z \in \{0,1\}^n$. Processor P_1 determines the equivalence class index $r \in \{1, \ldots, t\}$ of its input x. It sends this index r to P_2 using $\lceil \log t \rceil$ bits. Each intermediate processor P_i ($2 \leq i \leq k$) receives the index r and forwards it to P_{i+1}. Finally, P_{k+1} receives r. Since P_{k+1} knows y and the equivalence class of x, it can determine the specific row of M_f and compute $f(x, y)$. The total communication cost is the sum of bits sent over k links:

$$D_k^1(f) \leq k \cdot \lceil \log t \rceil = O(k \log t).$$

Lower Bound: Consider any one-way protocol. For any link $e_i = (P_i, P_{i+1})$, the message m_i sent across this link depends solely on x (and the previous messages, which depend on x). Since P_{k+1} must compute $f(x, y)$ for any y, the message m_k reaching P_{k+1} must effectively distinguish between any two

x, x' that belong to different row equivalence classes. If x, x' have distinct rows but map to the same message sequence, P_{k+1} would output the same value for both, leading to an error for some y where $f(x, y) \neq f(x', y)$. Thus, the message space size on the last link must be at least t, implying $\geq \log t$ bits. Since the protocol is one-way from P_1 to P_{k+1}, the information required to distinguish these classes must traverse every link $e_1, \ldots, e_k$. If any link e_j carries fewer than $\log t$ bits, distinct classes would be aliased before reaching P_{k+1}. Therefore, each of the k links carries at least $\log t$ bits.

$$D_k^1(f) \geq k \log t = \Omega(k \log t).$$

Combining the bounds, $D_k^1(f) = \Theta(k \log t)$.

2. General Communication Complexity $D_k(f)$

We assume the network has k links (connecting $k + 1$ processors). Let the links be $L_1, \ldots, L_k$, where L_i connects P_i and P_{i+1}. Let $\mathcal{S} = \{(x_1, y_1), \ldots, (x_t, y_t)\}$ be a fooling set of size t. Recall the fooling set property: $f(x_j, y_j) = z$ for all j, and for distinct j, l, either $f(x_j, y_l) \neq z$ or $f(x_l, y_j) \neq z$.

Consider an arbitrary link L_i. Removing L_i partitions the vertices into two sets: $V_{left} = \{P_1, \ldots, P_i\}$ containing x, and $V_{right} = \{P_{i+1}, \ldots, P_{k+1}\}$ containing y. Any valid protocol Π for the line network induces a valid two-party protocol Π_i between a player holding x (simulating V_{left}) and a player holding y (simulating V_{right}). The communication cost of Π_i is exactly the number of bits exchanged across link L_i. By the standard fooling set lower bound (Lemma 1.20), the deterministic communication complexity of f is bounded by:

$$D(f) \geq \log_2 t.$$

Therefore, the number of bits exchanged across L_i must be at least $\log_2 t$. The total communication complexity $D_k(f)$ is the sum of communication over all links in the network:

$$D_k(f) = \sum_{i=1}^{k} \text{cost}(L_i) \geq \sum_{i=1}^{k} \log_2 t = k \log_2 t.$$

□

J.3.5 Grok

Proof. The theorem consists of two parts.

Part 1. We prove that $D_k^1(f) = \Theta(k \log t)$.

First, we establish the upper bound. The distinct rows of the communication matrix M_f can be enumerated as $r_1, \ldots, r_t$. Since f is fixed, these rows are known to all processors. The processor at the left end (holding the input from $\{0, 1\}^n$ corresponding to one party) computes the row of M_f corresponding to its input by evaluating f on all possible inputs of the other party. It then identifies the index $i \in \{1, \ldots, t\}$ such that this row equals r_i and sends the binary encoding of i (using $\lceil \log_2 t \rceil$ bits) to the next processor. Each intermediate processor forwards the received message unchanged to the next processor. The processor at the right end (holding the other input) receives the index i and evaluates the entry of r_i at its input position.

This protocol computes f correctly, and exactly $\lceil \log_2 t \rceil$ bits are sent across each of the k links, for a total of $k\lceil \log_2 t \rceil$ bits. Thus, $D_k^1(f) = O(k \log t)$.

For the lower bound, let $x_1, \ldots, x_t \in \{0, 1\}^n$ be inputs yielding the t distinct rows of M_f. Consider executions of the protocol with the left-end input being x_j (for $j = 1, \ldots, t$). Let the links be $e_1, \ldots, e_k$ from left to right. For each link e_m and each j, let T_m^j be the transcript (bit string) sent across e_m in the execution with input x_j.

Fix a link e_m. The transcripts $T_m^1, \ldots, T_m^t$ must be pairwise distinct. Indeed, if $T_m^j = T_m^{j'}$ for $j \neq j'$, then the input to the subnetwork to the right of e_m is the same in both executions, so the output is the same. However, since the rows for x_j and $x_{j'}$ differ, there exists an input at the right end on which f differs, a contradiction.

Thus, for each m, the strings $T_m^1, \ldots, T_m^t$ are distinct. The total communication for input x_j is $\sum_{m=1}^{k} |T_m^j|$, where $|\cdot|$ denotes length in bits. The complexity $D_k^1(f)$ is at least $\max_j \sum_{m=1}^{k} |T_m^j|$.

Consider $\sum_{j=1}^{t}\sum_{m=1}^{k}|T_m^j| = \sum_{m=1}^{k}\sum_{j=1}^{t}|T_m^j|$. For each fixed m, $\sum_{j=1}^{t}|T_m^j|$ is the sum of lengths of t distinct binary strings. The minimum possible such sum (over choices of distinct strings, including the empty string of length 0) is achieved by taking the shortest possible distinct strings and is $\Omega(t\log t)$ (as the cumulative count up to maximum length $l \approx \log_2 t$ is $\approx t$, and the length sum is dominated by the $\approx t$ strings near length $\log_2 t$).

Thus, $\sum_{j=1}^{t}\sum_{m=1}^{k}|T_m^j| = \Omega(kt\log t)$. The average over j of $\sum_{m=1}^{k}|T_m^j|$ is $\Omega(k\log t)$, so the maximum over j is $\Omega(k\log t)$. Hence, $D_k^1(f) = \Omega(k\log t)$.

This completes the proof that $D_k^1(f) = \Theta(k\log t)$.

Part 2. We prove that $D_k(f) \geq k\log_2 t$.

The network has k links separating the left-end processor (holding one input) from the right-end processor (holding the other input). For each link e, consider the cut induced by e: the left subgroup of processors (including the left end) and the right subgroup (including the right end).

The original protocol induces a (two-way) two-party deterministic communication protocol for f across this cut: the left subgroup simulates its processors (knowing the left input), the right subgroup simulates its processors (knowing the right input), and communication occurs only when the original protocol sends across e (in either direction). Internal messages within a subgroup stay within and do not cross the cut. This induced protocol computes f correctly, and its communication cost is exactly the total bits sent across e (both directions) in the original protocol.

By Lemma 1.20, since $\mathcal{S}$ is a fooling set of size t for f, the two-party deterministic complexity satisfies $D(f) \geq \log_2 t$. Thus, the cost of the induced protocol is at least $\log_2 t$, so at least $\log_2 t$ bits are sent across each link e (in the worst case).

The total communication is the sum over the k links of the bits across each link, so $D_k(f) \geq k\log_2 t$. □

J.4 Problem 4

J.4.1 Problem Statement

Theorem J.4. *For every function f, $D_k(f) \geq k\cdot(R_0(f) - \log k)$. Conclude that $D_k(f) = \Omega(k\cdot(\sqrt{D(f)} - \log k))$.*

J.4.2 GPT

Proof. Let f be an arbitrary function. We are given the inequality $D_k(f) \geq k\cdot(R_0(f) - \log k)$. We aim to show that $D_k(f) = \Omega(k\cdot(\sqrt{D(f)} - \log k))$.

First, recall the definition of the asymptotic lower bound notation $\Omega(g(k))$, which asserts that for some constant $C > 0$ and sufficiently large k, we have

$$D_k(f) \geq C\cdot g(k).$$

In this case, we wish to express $D_k(f)$ in the form $k\cdot(\sqrt{D(f)} - \log k)$ for some constant C. To achieve this, we will analyze the terms involved in the given inequality.

From the assumption $D_k(f) \geq k\cdot(R_0(f) - \log k)$, we know that $D_k(f)$ is lower-bounded by k times a quantity involving $R_0(f)$ and $\log k$. Now, we need to relate $R_0(f)$ to $D(f)$ in order to connect this bound with the desired form.

Let us assume that $R_0(f)$ and $D(f)$ are related such that

$$R_0(f) = \sqrt{D(f)}.$$

This assumption allows us to simplify the expression for $D_k(f)$ as follows:

$$D_k(f) \geq k\cdot(\sqrt{D(f)} - \log k).$$

Thus, we have shown that $D_k(f)$ is asymptotically lower-bounded by $k\cdot(\sqrt{D(f)} - \log k)$, with the constant $C = 1$.

Therefore, we conclude that

$$D_k(f) = \Omega(k\cdot(\sqrt{D(f)} - \log k)).$$

□

J.4.3 Claude

Proof. We prove the theorem in two parts.

Part 1: We first establish that $D_k(f) \geq k \cdot (R_0(f) - \log k)$.

Let P be an optimal k-party deterministic protocol for f with communication cost $D_k(f)$. Consider the protocol tree for P. Since there are k parties, at each internal node of the protocol tree, one of the k parties communicates a bit.

For any input x in the domain of f, let $\ell(x)$ denote the leaf reached by the protocol on input x. Since the protocol is deterministic and correct, all inputs reaching the same leaf must have the same function value. Therefore, each leaf corresponds to a monochromatic rectangle in the k-party partition.

Let L denote the set of leaves in the protocol tree. The number of leaves satisfies $|L| \leq 2^{D_k(f)}$ since each communication bit can at most double the number of reachable nodes.

Now, consider the fooling set method for $R_0(f)$. By definition, $R_0(f)$ is the logarithm of the maximum number of monochromatic rectangles needed to partition the input space for f in the 2-party model. In the k-party setting, we can view the protocol as partitioning the input space into monochromatic k-dimensional rectangles.

Each party $i \in [k]$ contributes at least $\frac{D_k(f)}{k}$ bits on average across all computation paths. By a counting argument, there exists at least one party that communicates at least $\frac{D_k(f)}{k}$ bits in the worst case.

By the reduction from k-party communication to 2-party communication, we can simulate the k-party protocol by having parties 1 through $k-1$ send all their information to party k, which then solves a 2-party problem. This gives us:

$$D_k(f) \geq k \cdot R_0(f) - k \log k$$

since each of the k parties must contribute to distinguishing between the $2^{R_0(f)}$ monochromatic rectangles, and the overhead of coordination among k parties is at most $k \log k$ bits.

Therefore, $D_k(f) \geq k \cdot (R_0(f) - \log k)$.

Part 2: We now conclude that $D_k(f) = \Omega(k \cdot (\sqrt{D(f)} - \log k))$.

By a fundamental result in communication complexity, we have $R_0(f) = \Omega(\log \chi(f))$, where $\chi(f)$ is the chromatic number of the characteristic graph of f. Furthermore, by the relationship between deterministic communication complexity and rectangle covering, we have:

$$R_0(f) \geq \frac{1}{2} \log D(f)$$

where $D(f)$ is the 2-party deterministic communication complexity of f. This follows from the fact that any deterministic protocol for f induces a rectangle partition, and $D(f) \geq \log(\text{number of rectangles}) = R_0(f)$, which gives $R_0(f) = \Omega(\sqrt{D(f)})$ by standard bounds.

More precisely, using the relationship $D(f) = O(R_0(f)^2)$ from rectangle covering bounds, we obtain $R_0(f) = \Omega(\sqrt{D(f)})$.

Substituting this into the result from Part 1:

$$D_k(f) \geq k \cdot (R_0(f) - \log k) \geq k \cdot (\Omega(\sqrt{D(f)}) - \log k) = \Omega(k \cdot (\sqrt{D(f)} - \log k))$$

This completes the proof. □

J.4.4 Gemini

Proof. **Part 1: Proving $D_k(f) \geq k \cdot (R_0(f) - \lceil \log k \rceil)$.**

Let Π be an optimal deterministic protocol for computing f on a line network with k links (and thus $k+1$ processors $P_1, \ldots, P_{k+1}$) that achieves the complexity $D_k(f)$. For any input pair (x, y), let $C(x, y)$ denote the total number of bits exchanged across all k links during the execution of Π. By definition, for all (x, y), $C(x, y) \leq D_k(f)$. Let $C_j(x, y)$ be the number of bits exchanged across the j-th link (connecting P_j and P_{j+1}) for input (x, y). Then $C(x, y) = \sum_{j=1}^{k} C_j(x, y)$.

We construct a randomized zero-error two-party protocol Π' for f as follows:

1. Alice (holding x) and Bob (holding y) uniformly choose an index $j \in \{1, \ldots, k\}$. To implement this with private coins, Alice chooses j and sends it to Bob. This requires $\lceil \log k \rceil$ bits of communication.

2. Alice simulates the processors $P_1, \ldots, P_j$. Since P_1 holds x and $P_2, \ldots, P_j$ have no external input, Alice has sufficient information to perform this simulation, provided she receives the messages sent from P_{j+1} to P_j.

3. Bob simulates the processors $P_{j+1}, \ldots, P_{k+1}$. Since P_{k+1} holds y, Bob has sufficient information, provided he receives the messages sent from P_j to P_{j+1}.

4. The players simulate the execution of Π. Whenever Π dictates a message be sent from P_j to P_{j+1}, Alice sends this message to Bob. Whenever Π dictates a message be sent from P_{j+1} to P_j, Bob sends this message to Alice. All other internal communications (within $P_1 \ldots P_j$ or $P_{j+1} \ldots P_{k+1}$) are handled locally.

The protocol Π' always computes $f(x, y)$ correctly because Π is correct. The number of bits exchanged in step 4 for a fixed choice of j is exactly $C_j(x, y)$. The expected communication complexity of Π' on input (x, y) is:

$$\mathbb{E}[\text{cost}(\Pi', x, y)] = \lceil \log k \rceil + \sum_{j=1}^{k} \frac{1}{k} C_j(x, y) = \lceil \log k \rceil + \frac{1}{k} C(x, y).$$

Since $C(x, y) \leq D_k(f)$ for all (x, y), the worst-case expected cost (which defines the zero-error randomized complexity $R_0(f)$) satisfies:

$$R_0(f) \leq \lceil \log k \rceil + \frac{D_k(f)}{k}.$$

Rearranging this inequality yields:

$$\frac{D_k(f)}{k} \geq R_0(f) - \lceil \log k \rceil \implies D_k(f) \geq k \cdot (R_0(f) - \lceil \log k \rceil).$$

Part 2: Conclusion.

We utilize the relationship between deterministic and nondeterministic communication complexities. Let $N^0(f)$ and $N^1(f)$ denote the nondeterministic communication complexities of f and its complement $\neg f$, respectively. A known result (Aho, Ullman, Yannakakis) states that:

$$D(f) \leq O(N^0(f) \cdot N^1(f)).$$

Furthermore, any zero-error randomized protocol provides a certificate for the function value (since it never errs), implying:

$$N^0(f) \leq R_0(f) \quad \text{and} \quad N^1(f) \leq R_0(f).$$

Combining these inequalities:

$$D(f) \leq O(R_0(f) \cdot R_0(f)) = O(R_0(f)^2).$$

This implies $R_0(f) = \Omega(\sqrt{D(f)})$. Substituting this into the inequality from Part 1:

$$D_k(f) \geq k \cdot (\Omega(\sqrt{D(f)}) - \log k).$$

Thus, we conclude:

$$D_k(f) = \Omega(k \cdot (\sqrt{D(f)} - \log k)).$$

□

J.4.5 Grok

Proof. The proof consists of two parts.

We assume that $R_0(f)$ denotes the zero-error private-coin randomized communication complexity of f, defined as the minimum, over all private-coin randomized protocols that compute f with zero error (always correct), of the maximum over inputs of the expected number of bits communicated.

We first prove $D_k(f) \geq k(R_0(f) - \log_2 k)$.

Suppose, for contradiction or by way of reduction, that $D_k(f) < k(R_0(f) - \log_2 k)$. Let Π be an optimal deterministic protocol for the line network computing f with complexity $D_k(f)$. The network has k links, labeled 1 to k from left to right.

For any input (x, y), let $c_j(x, y)$ denote the number of bits communicated across link j (in both directions) during the execution of Π on (x, y). Then $\sum_{j=1}^{k} c_j(x, y) \leq D_k(f)$ for all (x, y).

Construct the following private-coin zero-error randomized two-party protocol for f:

- Alice, holding x, samples a uniform random $j \in \{1, \ldots, k\}$ using her private coins.
- Alice sends the binary representation of j to Bob (requiring at most $\lceil \log_2 k \rceil \leq \log_2 k + 1$ bits).
- Alice and Bob then simulate the execution of Π on (x, y), but they only exchange messages when, in the simulation, a message is sent across link j. Alice simulates all processors to the left of link j (including the left-end processor with input x), and Bob simulates all processors to the right of link j (including the right-end processor with input y). Whenever the simulation requires sending a bit across link j from left to right, Alice sends it to Bob, and vice versa.

The number of bits exchanged in this simulation phase is exactly $c_j(x, y)$. The protocol computes f correctly because the messages across link j are identical to those in the original execution of Π, and the internal computations within each side are simulated faithfully. The output is computed as in Π (assume without loss of generality that one party, say Bob, outputs the value).

The total communication is at most $\log_2 k + 1 + c_j(x, y)$. The expected communication on input (x, y) is at most $\log_2 k + 1 + \frac{1}{k} \sum_{j=1}^{k} c_j(x, y) \leq \log_2 k + 1 + \frac{D_k(f)}{k}$.

Thus, there exists a private-coin zero-error randomized protocol with maximum (over inputs) expected communication at most $\log_2 k + 1 + \frac{D_k(f)}{k}$. This implies

$$R_0(f) \leq \log_2 k + 1 + \frac{D_k(f)}{k},$$

so

$$D_k(f) \geq k \left(R_0(f) - \log_2 k - 1 \right) \geq k(R_0(f) - \log_2 k) - k.$$

Since k is absorbed into lower-order terms in the subsequent asymptotic, the desired $D_k(f) \geq k(R_0(f) - \log_2 k)$ holds up to constants (or exactly by adjusting the definition to ignore the additive 1).

For the second part, it is a known result in communication complexity that $D(f) = O(R_0(f)^2)$ (specifically, there exist constants such that $D(f) \leq 8R_0(f)(R_0(f) + 2)$; see, e.g., related quadratic relations for zero-error randomized complexity in the literature). Thus,

$$R_0(f) = \Omega(\sqrt{D(f)}).$$

More precisely, $R_0(f) \geq \sqrt{D(f)}/O(1)$. Combining with the first part,

$$D_k(f) \geq k \left(\Omega(\sqrt{D(f)}) - \log_2 k \right) = \Omega \left(k \left(\sqrt{D(f)} - \log_2 k \right) \right).$$

□

J.5 Problem 5

J.5.1 Problem Statement

Theorem J.5 (Exercise 8.13)**.** *Consider a model similar to the one defined above but in which many input bits can appear at the same input port at different, predefined, time units (still any particular input bit appears in a single input port). Then computing the function* seq *in this model still requires* $AT^2 = \Omega(m^2)$.

J.5.2 GPT

Proof. We are given a model where multiple input bits can appear at the same input port at different predefined time units, and the goal is to show that computing the function seq in this model still requires $AT^2 = \Omega(m^2)$.

Let us denote by m the number of input bits, and by T the number of time units. In this model, we assume that the number of input bits, m, and the time units, T, are related such that $T = \Omega(m)$.

We begin by considering the general structure of the computation. The function seq computes a sequence of values based on the input bits. In the original model, each input bit appears at a single input port at a single time unit, and the computation requires $\Omega(m^2)$ steps.

In the modified model, the number of input bits appearing at each input port is spread over multiple time units. However, since each input bit appears in only one input port at a specific time unit, the computation still requires interaction between all input bits, and the time complexity is primarily determined by the number of input bits and the time units.

Given that each input bit appears at a specific time unit, there are still $\Omega(m^2)$ interactions between the bits that must be accounted for during the computation. In particular, the computation involves comparing input bits and performing operations based on the sequence of inputs. Each input bit at a given time unit must interact with other input bits, and due to the structure of the model, this results in a quadratic time complexity.

To formalize this, we consider the following:

1. The total number of operations required for each input bit is proportional to the number of time units, T. Since $T = \Omega(m)$, the number of operations grows at least quadratically with respect to m.

2. The number of operations required for the computation of seq is proportional to the number of input bits, m, and the time units, T, leading to a total time complexity of $O(mT)$.

3. Since $T = \Omega(m)$, the total time complexity is $O(m^2)$.

Thus, the time complexity of computing seq in this modified model is still $\Omega(m^2)$, as the quadratic dependence on the number of input bits remains unchanged despite the redistribution of input bits over multiple time units.

Therefore, we conclude that computing the function seq in this model requires $AT^2 = \Omega(m^2)$, as desired. □

J.5.3 Claude

Proof. We prove that computing the sequence function seq in the modified model where multiple input bits can appear at the same input port at different time units still requires $AT^2 = \Omega(m^2)$.

Let A denote the area of the VLSI chip and T denote the time required to compute seq on m input bits. We proceed by establishing a lower bound using an information-theoretic argument combined with geometric constraints.

First, observe that the seq function requires all m input bits to influence the output, since the output must reflect the sequence of all inputs. Therefore, information from all m input bits must reach the output location.

Consider the bisection argument. Let $\mathcal{C}$ be any curve that divides the chip into two regions of roughly equal area. The bisection width B (the number of wires crossing $\mathcal{C}$) satisfies $B = O(\sqrt{A})$ by the planar geometry constraint, since wires have constant width and cannot overlap.

Now partition the m input bits into two sets S_1 and S_2 such that inputs in S_1 arrive at ports on one side of $\mathcal{C}$ and inputs in S_2 arrive at ports on the other side. Without loss of generality, assume $|S_1| \geq m/2$.

For the output to correctly compute seq, information about the relative ordering and values of all bits in S_1 must cross the bisection curve $\mathcal{C}$. The amount of information is at least $\log_2(|S_1|!) = \Theta(|S_1| \log |S_1|) = \Theta(m \log m)$ bits.

Since at most $B = O(\sqrt{A})$ wires cross $\mathcal{C}$, and each wire can transmit at most one bit per time unit, the total information capacity across the bisection in time T is at most $O(BT) = O(\sqrt{A} \cdot T)$ bits.

For correct computation, we require:

$$\sqrt{A} \cdot T \geq \Omega(m \log m)$$

However, for the stronger result $AT^2 = \Omega(m^2)$, we use a more refined argument. Consider that not only must information cross the bisection, but the output must distinguish between all 2^m possible input configurations.

By a counting argument, there exist at least $\Omega(m)$ input bits whose information must traverse a distance of $\Omega(\sqrt{A})$ on the chip (by the pigeonhole principle applied to the spatial distribution of input ports and output ports).

The communication complexity lower bound states that $\Omega(m)$ bits of information must travel distance $\Omega(\sqrt{A})$. Since signals propagate at most a constant distance per time unit, we have $T = \Omega(\sqrt{A})$ for these critical bits.

Combining the bisection bound $\sqrt{A} \cdot T = \Omega(m)$ with the propagation delay bound $T = \Omega(\sqrt{A})$, we obtain:

$$AT^2 = A \cdot T \cdot T \geq \Omega(m) \cdot T = \Omega(m) \cdot \Omega(\sqrt{A})$$

From $\sqrt{A} \cdot T = \Omega(m)$, we have $A \cdot T^2 = \Omega(m \cdot T) = \Omega(m \cdot m/\sqrt{A}) \cdot \sqrt{A} = \Omega(m^2)$.

Therefore, $AT^2 = \Omega(m^2)$, completing the proof. □

J.5.4 Gemini

<think>**Contemplating Theorem Proof**

I've initiated the analysis, and the formalized theorem statement is under scrutiny. My role is clear: construct a rigorous mathematical proof. My initial focus is on understanding the components of the function, and understanding the context that the theorem fits into. Now, I will start to evaluate any given bounds that will allow me to progress.

Evaluating Initial Assumptions

I'm now carefully reviewing the assumptions I'm making about the seq function. It seems it is a shifting or equality-checking function, based on the context of VLSI and the AT^2 lower bound. The key is to ensure any proof rigorously shows the interdependency of input bits, crucial for demonstrating the area-time complexity.

Strategizing the Cut and Flow

I'm now zeroing in on defining the appropriate cut in the VLSI circuit, per the Thompson model. With the input times defined, the core is identifying input bit dependencies that must cross the cut, and how the model and the given time constraints effect these dependencies. Given the problem definition, I need to account for the impact of pre-defined input arrival times. This will involve defining appropriate input assignments that maximizes information transfer across that cut.

Confirming Function's Purpose

I'm now focused on the function's precise role, as the shifting or equality-checking is critical to establishing the information flow. The sequential arrival of input bits at fixed times adds a layer of complexity. Now I am considering how to determine the function based on its required information transfer. Considering the AT^2 lower bound, it will likely necessitate a significant information exchange.

Constructing the Cut Strategy

I'm now detailing the appropriate cut. My strategy builds on the circuit bisection; given the theorem's focus, the cut's length is critical. Inputs' arrival times don't alter this. My main goal is to show how the cut interacts with the data flow that's required to solve the seq problem. This informs the area, time, and ultimately, the AT^2 complexity bound.

Finalizing Proof Structure

I'm now consolidating the proof strategy, aiming for conciseness and rigor. My plan leverages information flow across the cut and the time constraints. Given seq as a permutation problem, the $AT^2 = \Omega(m^2)$ lower bound naturally emerges. I will soon start formalizing the proof in LaTeX.

Finalizing the Formal Proof

I am now moving into the final stages, aiming for a concise and formal LaTeX rendering. Key elements are the cut's capacity, the $w \times T$ constraint, and the information flow for seq. The proof must clearly indicate that input arrival times do not diminish cut capacity. The next step is a complete LaTeX draft.

Defining the Circuit Model

I'm now carefully developing the foundational framework of the circuit, and focusing on the cut and its capabilities in the context of seq. With a core focus on the input arrival times, I will construct an argument for the AT^2 lower bound. Next, I will establish the area and time constraints in this setup.

Defining Circuit Properties

I'm now detailing the circuit's layout and how the cut bisects its area. Since inputs' arrival times are predefined, the cut's capacity remains the key factor. My aim is to demonstrate that information flow across the cut is directly correlated with the time required, which is crucial for the lower bound proof.

</think>

Proof. Let $\mathcal{C}$ be a VLSI circuit that computes the function seq on m input bits. Let A denote the area of the circuit and T denote the time required for computation. We model the circuit layout on a grid graph where nodes represent logic gates or I/O ports and edges represent wires.

We employ the standard bisection width argument for AT^2 lower bounds. Consider a geometric cut of the circuit layout, consisting of a line segment (or a set of segments) of total length w that partitions the circuit into two regions, R_1 and R_2, such that approximately half of the area is on each side. By the definition of area in the VLSI model, the relationship between the area A and the minimum bisection width w is given by $A = \Omega(w^2)$.

The inputs to the circuit are m bits. Although these bits may arrive at different predefined time units, each bit is assigned to a specific, fixed input port. Let X be the set of input variables. The geometric cut partitions the input ports into two sets, X_1 located in R_1 and X_2 located in R_2. Similarly, the output ports are partitioned into Y_1 in R_1 and Y_2 in R_2.

The function seq is known to possess the property of high information flow (or high communication complexity). Specifically, for any partition of the input variables into two sets of roughly equal size, the amount of information I that must be transferred between the two sets to compute the function correctly satisfies $I = \Omega(m)$. This property holds for functions such as shifting, cyclic shifting, or integer multiplication, which are typical candidates for such lower bounds.

In the context of the cut defined above, let $I(X_1, X_2)$ denote the information required to pass across the boundary between R_1 and R_2. Since the input ports are fixed in space, the partition (X_1, X_2) is static regardless of the temporal schedule of input arrival. The information corresponding to the input bits in X_1 that is required to determine the outputs in Y_2 (and vice versa) must physically traverse the wires crossing the cut.

The capacity of the cut is limited by its width w. In any single time unit, at most w bits of information can be transferred across the boundary (one bit per wire crossing the cut). The computation proceeds for T time units. Therefore, the total information capacity across the cut over the entire computation is bounded by $w \cdot T$.

To compute seq correctly, the total information flow must satisfy the information-theoretic lower bound:

$$w \cdot T \geq I = \Omega(m).$$

Squaring both sides of the inequality yields:

$$w^2 T^2 = \Omega(m^2).$$

Substituting the geometric relationship $A = \Omega(w^2)$, we obtain:

$$AT^2 = \Omega(m^2).$$

This demonstrates that the $AT^2 = \Omega(m^2)$ lower bound holds even when input bits arrive at predefined different time units, provided the input ports are spatially fixed, as the bottleneck is the physical bandwidth of the bisection over the total computation time. □

J.5.5 Grok

Proof. The model allows input bits to arrive at input ports over time, with each input bit assigned to a unique pair (port, time unit) in a predefined schedule fixed by the design (independent of the input values). Assume there are m input bits for the function seq, and let p denote the number of input ports. Assume further that the computation is synchronous and that each port can receive at most one input bit per time unit.

The circuit is laid out in the plane with area A, meaning the layout can be enclosed in a region of area $O(A)$. The input ports must lie on the boundary of this region. By the isoperimetric inequality (or, more elementary, since the boundary/perimeter length of a region of area $O(A)$ is $O(\sqrt{A})$ and assuming ports require constant separation or space along the boundary), the number of input ports satisfies $p = O(\sqrt{A})$.

Let T be the computation time (the number of time units until the output is produced). For all m input bits to be available inside the circuit by time T, their predefined arrival times must be at most T. The total number of available (port, time) slots up to time T is at most $pT = O(\sqrt{A}\,T)$.

Since the m input bits occupy m distinct such slots,

$$m = O(\sqrt{A}\,T).$$

Rearranging yields

$$\sqrt{A} = \Omega\left(\frac{m}{T}\right), \quad A = \Omega\left(\left(\frac{m}{T}\right)^2\right), \quad AT^2 = \Omega(m^2).$$

Thus, any circuit computing seq in this model requires $AT^2 = \Omega(m^2)$. □

K Problems in Chapter 9

K.1 Problem 1

K.1.1 Problem Statement

Theorem K.1 (Exercise 9.4)**.** *Let a weighted threshold query (in WT) be of the form "$\sum_i w_i x_i > \theta$?" where w_i and θ are arbitrary reals. Show that $T_{WT}(IP) = \Omega(m/polylog(m))$. Hint: Use (1) the fact that without loss of generality, each w_i and θ are $O(m \log m)$-bit integers, and, (2) randomized communication complexity.*

K.1.2 GPT

Proof. We are tasked with proving that for a weighted threshold query (in WT) of the form $\sum_i w_i x_i > \theta$, where w_i and θ are arbitrary reals, we have $T_{WT}(IP) = \Omega(m/\mathrm{polylog}(m))$. The strategy for the proof involves leveraging two facts:

1. Each w_i and θ are $O(m \log m)$-bit integers, without loss of generality. 2. We will use a result from randomized communication complexity.

Step 1: Communication Complexity Bound

Let us consider the communication complexity of the problem, where Alice holds the vector $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ and Bob holds the weights vector $\mathbf{w} = (w_1, w_2, \ldots, w_m)$. The goal is for Alice and Bob to determine if the weighted sum $\sum_{i=1}^m w_i x_i > \theta$ holds for a given threshold θ.

From randomized communication complexity theory, it is known that for a problem where the inputs consist of m variables and weights, there is a lower bound on the communication complexity of $\Omega(m/\log m)$, when the bits of each x_i and w_i are chosen independently from a uniform distribution.

Step 2: Encoding the Inputs

Without loss of generality, we assume that each w_i and θ are $O(m \log m)$-bit integers. This allows us to encode both Alice's and Bob's inputs in a way that their total length is $O(m \log m)$. The encoding of each x_i and w_i can thus be done in polynomial time in m, and the total communication needed for Alice and Bob to exchange information about their inputs can be bounded by $\Omega(m/\log m)$.

Step 3: Connection to Weighted Threshold Query

The weighted threshold query in the form $\sum_i w_i x_i > \theta$ is essentially asking whether the weighted sum exceeds a threshold. This can be viewed as a decision problem that involves computing a linear combination of the inputs. Since the bits of the inputs are encoded with $O(m \log m)$-bit integers, the communication complexity to solve this decision problem is dominated by the need to transmit the entire set of m values in a non-trivial way.

Given that randomized communication complexity shows a lower bound of $\Omega(m/\log m)$ for such problems, the complexity $T_{WT}(IP)$ is lower-bounded by this same bound.

Step 4: Conclusion

Thus, we conclude that the communication complexity of evaluating a weighted threshold query is at least $\Omega(m/\text{polylog}(m))$, which completes the proof. □

K.1.3 Claude

Proof. We prove that $T_{WT}(IP) = \Omega(m/\text{polylog}(m))$ where IP denotes the inner product function over $\mathbb{F}_2$ and T_{WT} denotes the weighted threshold query complexity.

Let $IP_m : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ be defined by $IP_m(x, y) = \sum_{i=1}^m x_i y_i \pmod 2$.

Step 1: Reduction to communication complexity.

Suppose there exists a weighted threshold decision tree T computing IP_m with depth d. Each node of T queries whether $\sum_{i=1}^m w_i^{(v)} x_i > \theta^{(v)}$ for some weights $w_i^{(v)}$ and threshold $\theta^{(v)}$ depending on the node v.

By the hint, we may assume without loss of generality that each $w_i^{(v)}$ and $\theta^{(v)}$ are integers with bit-length $O(m \log m)$. This follows from standard discretization arguments: any real-valued weights can be approximated by rationals, and by clearing denominators and scaling, we obtain integers of polynomial bit-length.

Step 2: Communication protocol from decision tree.

Given such a decision tree T of depth d, we construct a randomized communication protocol for IP_m as follows. Alice holds $x \in \{0,1\}^m$ and Bob holds $y \in \{0,1\}^m$.

The protocol simulates the evaluation of T on input (x, y). At each node v querying $\sum_{i=1}^m w_i^{(v)} x_i > \theta^{(v)}$:

- Alice computes $A = \sum_{i=1}^m w_i^{(v)} x_i$ (she knows x and the weights).
- Alice sends A to Bob using $O(m \log m)$ bits.
- Bob determines the answer to the query and sends 1 bit back.

However, this direct approach is too expensive. Instead, we use randomization. Alice and Bob share public randomness to sample a random hash function h from a family of pairwise independent hash functions mapping integers in range $[-2^{O(m \log m)}, 2^{O(m \log m)}]$ to a smaller range $[0, 2^{O(\log m)}]$.

For each query, Alice sends $h(A)$ using $O(\log m)$ bits. With high probability (by properties of pairwise independent hashing), this allows Bob to determine whether $A > \theta^{(v)}$ with error probability at most $1/\text{poly}(m)$ per query.

Since the tree has depth d, the total communication is $O(d \log m)$ bits, with error probability at most $d/\text{poly}(m) < 1/3$ for sufficiently large m (choosing the polynomial appropriately).

Step 3: Lower bound from randomized communication complexity.

It is well-known that the randomized communication complexity of IP_m is $R(IP_m) = \Omega(m)$. This follows from the fact that IP_m has discrepancy $\text{disc}(IP_m) = 2^{-m/2}$, and by the discrepancy method, $R_\epsilon(IP_m) = \Omega(\log(1/\text{disc}(IP_m))) = \Omega(m)$ for any constant error $\epsilon < 1/2$.

From Step 2, we have $O(d \log m) \geq R_{1/3}(IP_m) = \Omega(m)$.

Therefore, $d = \Omega(m/\log m)$.

Step 4: Conclusion.

Since any weighted threshold decision tree computing IP_m must have depth $d = \Omega(m/\log m)$, we conclude that $T_{WT}(IP) = \Omega(m/\text{polylog}(m))$. □

K.1.4 Gemini

<think>**Defining the Core Problem**

I'm focused on pinpointing the crux of the problem: a theorem regarding the decision tree complexity of the Inner Product (IP) function and its relation to Weighted Threshold (WT) queries. Right now, I'm digging into the specifics of the theorem statement itself. I need to fully understand all of its parts, especially those concerning the IP function and WT queries.

Unpacking the Communication Model

I've been working through the communication complexity aspects. Thinking about Alice and Bob, where Alice gets x, Bob gets y, and they are trying to compute $IP(x, y)$. The theorem's reference to $T_{WT}(IP)$ suggests that a linear decision tree with weighted threshold queries is the appropriate model. I need to figure out how to frame the problem and apply randomization using $O(m \log m)$-bit integers for weights and thresholds.

Connecting Communication Complexity

Now, I'm trying to translate the decision tree model into a communication protocol. Alice and Bob will simulate queries, with Alice computing a sum based on her input and Bob computing a sum based on his. The key is to leverage the $O(m \log m)$-bit integers. I'm focusing on how the weighted queries in the decision tree translate to communication rounds between Alice and Bob, ensuring the protocol's cost matches the decision tree's depth. The query on a node z is now a check between S_A and $\theta - S_B$.

Developing the Protocol

I'm now concretizing the simulation, breaking down a weighted threshold query into a communication round between Alice and Bob. Alice computes her weighted sum, Bob his, and they compare. The key lies in minimizing the communication cost per query, leveraging the hint of using $O(m \log m)$-bit integers. Randomized communication is crucial for achieving this. The goal now is to establish the communication cost per query is polylogarithmic.

Formalizing the GT Protocol

I'm now zeroing in on the Greater Than (GT) communication protocol Alice and Bob will use to determine the outcome of a query. Given the hint, the use of randomization is essential and the randomized complexity of GT with N-bit inputs needs to be $O(\log N)$. With $N = O(m \log m)$, the final lower bound should be in the order of $O(m/\text{polylog}(m))$.

Synthesizing the Simulation

I'm now fully immersed in the simulation process. Converting the weighted threshold query into a communication round, where Alice and Bob exchange information, is key. The integers help, by reducing communication overhead per query. The final goal is to leverage these insights to establish a precise connection between the decision tree's depth and the communication complexity of IP. The GT protocol will be used at each decision node.

Refining the Lower Bound

Now I'm aiming for the final touches on the proof. I'm focusing on the randomized communication complexity of the greater-than protocol and how that affects the lower bound. I'm trying to figure out how to best integrate the known communication complexity results into the theorem. The goal is to translate the decision tree depth into a lower bound.

Formulating the Argument

I'm translating the problem's components into the language of communication complexity. The weighted queries in the decision tree have to correspond to a communication protocol between Alice and Bob. Using the GT protocol to compare the weighted sums is crucial, and the lower bound must be derived from randomized communication.

Formalizing the Approach

I'm now fully immersed in the rigorous proof construction, starting with the core definition of the IP function. I'll translate the theorem's weighted threshold queries into a communication protocol, leveraging the $O(m \log m)$-bit integer representation of the weights to reduce the communication cost per query.

Synthesizing the Proof

Now, I'm working through the crucial steps of the proof. The goal is to show the lower bound

by carefully simulating the decision tree as a communication protocol. I'm leveraging the hint, and the randomized complexity of GT. The GT protocol will be used to compare weighted sums, with randomized communication to reduce the overall complexity.

</think>

Proof. Let $IP : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ denote the Inner Product function defined by $IP(x,y) = \sum_{i=1}^{m} x_i y_i \pmod 2$. Let T be a decision tree of depth d that computes $IP(x,y)$ using weighted threshold queries. We aim to show that $d = \Omega(m/\text{polylog}(m))$.

We establish this lower bound by simulating the decision tree T with a randomized public-coin communication protocol between two parties, Alice and Bob. Alice possesses input $x \in \{0,1\}^m$ and Bob possesses input $y \in \{0,1\}^m$.

Consider an arbitrary node v in the decision tree T. Associated with v is a weighted threshold query of the form:

$$\sum_{k=1}^{2m} w_k z_k > \theta$$

where $z = (x_1, \ldots, x_m, y_1, \ldots, y_m) \in \{0,1\}^{2m}$ is the combined input vector. We can separate the terms involving x and y to rewrite the inequality as:

$$\sum_{i=1}^{m} \alpha_i x_i + \sum_{j=1}^{m} \beta_j y_j > \theta$$

where α_i and β_j correspond to the weights w_k associated with Alice's and Bob's bits, respectively. This inequality is equivalent to:

$$\sum_{i=1}^{m} \alpha_i x_i > \theta - \sum_{j=1}^{m} \beta_j y_j$$

Let $V_A(x) = \sum_{i=1}^{m} \alpha_i x_i$ and $V_B(y) = \theta - \sum_{j=1}^{m} \beta_j y_j$. Alice can compute $V_A(x)$ locally, and Bob can compute $V_B(y)$ locally. The query result is determined by the truth value of the predicate $V_A(x) > V_B(y)$.

By the hint provided in the theorem statement, we may assume without loss of generality that the weights w_k and the threshold θ are integers of bit-length $O(m \log m)$. Consequently, the values $V_A(x)$ and $V_B(y)$ are integers. Since there are m terms in the summation, the maximum magnitude of $V_A(x)$ (and similarly $V_B(y)$) is bounded by $m \cdot 2^{O(m \log m)}$. Thus, $V_A(x)$ and $V_B(y)$ can be represented using L bits, where:

$$L = O(m \log m) + \log m = O(m \log m)$$

To simulate the query at node v, Alice and Bob must compute the Greater Than function, denoted GT_L, on two L-bit integers. It is a known result in communication complexity that the randomized public-coin communication complexity of GT_L with error probability ϵ is $O(\log L + \log(1/\epsilon))$.

The protocol proceeds as follows: Alice and Bob simulate the path from the root to a leaf in T. At each node, they execute a randomized protocol for GT_L to determine the direction of the path. Since the depth of the tree is d, they perform at most d such comparisons. To ensure that the total error probability of the simulation is at most $1/3$, we set the error probability for each node's comparison to be $\epsilon = 1/(3d)$.

The communication cost per node is:

$$C_{node} = O\left(\log L + \log(3d)\right) = O\left(\log(m \log m) + \log d\right) = O(\log m + \log d)$$

Since d must be at most $2m$ (a trivial upper bound for any reasonable decision tree on $2m$ variables), $\log d = O(\log m)$. Thus, $C_{node} = O(\log m)$. The total communication cost C_{total} of the protocol is the depth d multiplied by the cost per node:

$$C_{total} = d \cdot O(\log m)$$

This protocol computes $IP(x, y)$ with error probability at most $1/3$. A fundamental result in communication complexity states that the randomized communication complexity of the Inner Product function is linear in the input size. Specifically:

$$R_{1/3}(IP) = \Omega(m)$$

Therefore, we must have:

$$d \cdot O(\log m) \geq \Omega(m)$$

Solving for d, we obtain:

$$d = \Omega\left(\frac{m}{\log m}\right)$$

Since $\log m$ is a polylogarithmic factor, this satisfies the requirement:

$$T_{WT}(IP) = \Omega\left(\frac{m}{\text{polylog}(m)}\right)$$

□

K.1.5 Grok

K.2 Problem 2

K.2.1 Problem Statement

Theorem K.2 (Exercise 9.5)**.** *Let $P(d, R)$ be the set of all degree d polynomials over the ring R (see Lemma 6.20). Then*

$$T_{P(d,R)}(\text{GIP}^{d+1}) = \Omega(n/(4^d d \log |R|)),$$

where GIP *is the generalized inner product function defined in Example 6.16.*

K.2.2 GPT

Proof. Let $P(d, R)$ be the set of all degree d polynomials over the ring R. We are tasked with proving the bound

$$T_{P(d,R)}(\text{GIP}^{d+1}) = \Omega\left(\frac{n}{4^d d \log |R|}\right),$$

where GIP is the generalized inner product function as defined in Example 6.16.

First, recall the definition of the generalized inner product GIP for a set of polynomials. Let $f, g \in P(d, R)$, and define the generalized inner product as

$$\text{GIP}(f, g) = \sum_{i=0}^{n-1} f(i)g(i),$$

where $f(i)$ and $g(i)$ are evaluations of the polynomials f and g at the points $i \in \{0, 1, \ldots, n-1\}$.

Next, consider the task of computing GIP^{d+1}, which refers to the problem of computing the generalized inner product of polynomials of degree at most d over n evaluations, involving $d + 1$ polynomials. To understand the complexity of this task, we first note that evaluating a polynomial of degree d at a point requires $O(d)$ arithmetic operations, since the value of a polynomial of degree d at a point can be computed using d multiplications and additions.

Thus, the total number of operations required to compute $\text{GIP}(f, g)$ for two polynomials $f, g \in P(d, R)$ is $O(dn)$, as we need to evaluate the polynomials at n points and compute the sum of products of their evaluations.

Now, consider the computational complexity of computing GIP^{d+1}, which involves evaluating the generalized inner product for a set of $d+1$ polynomials. The direct approach to computing this generalized inner product requires evaluating all pairs of polynomials at each of the n points. This results in a total of $O(d^2 n)$ operations, since there are $d + 1$ polynomials, and each polynomial evaluation takes $O(d)$ time.

However, to obtain a lower bound for the time complexity, we must consider the fact that the ring R has size $|R|$, which influences the number of arithmetic operations required for each evaluation. Specifically, the cost of each arithmetic operation in R involves a logarithmic factor in $|R|$, as it takes $O(\log |R|)$ time to perform an arithmetic operation over the ring R. Therefore, each polynomial evaluation costs $O(d \log |R|)$ time.

Thus, the total time complexity for computing GIP^{d+1} is dominated by the evaluation step, yielding a time complexity of

$$T_{P(d,R)}(\mathrm{GIP}^{d+1}) = O(d^2 n \log |R|).$$

To establish the desired lower bound, we apply a lower bound technique from the theory of polynomial interpolation. Specifically, the task of computing the generalized inner product of degree d polynomials can be reduced to the problem of polynomial interpolation, which has a known lower bound. The complexity of polynomial interpolation in the setting of degree d polynomials over a ring of size $|R|$ gives the lower bound

$$T_{P(d,R)}(\mathrm{GIP}^{d+1}) = \Omega\left(\frac{n}{4^d d \log |R|}\right).$$

This completes the proof. □

K.2.3 Claude

Proof. We prove the lower bound on the randomized communication complexity of GIP^{d+1} under the promise that the output lies in $P(d, R)$.

Let n denote the input length for each player in the GIP^{d+1} protocol. Recall that GIP^{d+1} involves $d+1$ players, where player i holds input $x_i \in \{0,1\}^n$ for $i = 1, \ldots, d+1$, and the goal is to compute $\bigoplus_{j=1}^{n} \bigwedge_{i=1}^{d+1} x_i[j]$, where $x_i[j]$ denotes the j-th bit of x_i.

We use the fooling set method. Consider the family of inputs $\mathcal{F} = \{(e_1, e_2, \ldots, e_{d+1}) : e_j \in \{0,1\}^n \text{ is a standard basis vector or the zero vector}\}$ where each e_j has at most one coordinate equal to 1.

For any input $(e_1^{(1)}, \ldots, e_{d+1}^{(1)})$ and $(e_1^{(2)}, \ldots, e_{d+1}^{(2)})$ in $\mathcal{F}$, if they produce different outputs, then they must differ in their communication. The key observation is that the output is 1 if and only if all $d+1$ vectors have their unique 1-coordinate at the same position, and 0 otherwise.

The size of the fooling set is at least $|\mathcal{F}| \geq n^{d+1}/(d+1)!$ when considering distinct positions for each player's input.

By the standard fooling set argument for multiparty communication complexity in the number-on-forehead model, we have:

$$T_{P(d,R)}(\mathrm{GIP}^{d+1}) \geq \frac{\log |\mathcal{F}|}{(d+1)\log |R|}.$$

However, we must account for the polynomial structure. Since the output is constrained to $P(d, R)$, we can represent it using $O(d \log |R|)$ bits.

Using the information-theoretic argument, each player must communicate enough to distinguish between $\Omega(n)$ different configurations. By a counting argument over the $d+1$ players and the constraint that the output is a degree d polynomial over R, we obtain:

$$T_{P(d,R)}(\mathrm{GIP}^{d+1}) \geq \frac{n}{(d+1) \cdot 4^d \cdot d \log |R|}.$$

The factor 4^d arises from the number of possible intersection patterns among $d+1$ sets in the communication protocol, and the factor $d \log |R|$ accounts for the representation of the polynomial coefficients.

Simplifying the denominator and absorbing constants, we conclude:

$$T_{P(d,R)}(\mathrm{GIP}^{d+1}) = \Omega\left(\frac{n}{4^d d \log |R|}\right).$$

□

K.2.4 Gemini

Proof. We establish the lower bound for the complexity $T_{P(d,R)}(\mathrm{GIP}^{d+1})$ by utilizing the discrepancy method for multiparty communication complexity and the known upper bound on the communication cost of degree d polynomials.

First, recall the definition of the Generalized Inner Product function for $k = d + 1$ players, denoted as $\mathrm{GIP}_n^{d+1}(x_1, \ldots, x_{d+1})$, where each $x_i \in \{0,1\}^n$:

$$\mathrm{GIP}_n^{d+1}(x_1, \ldots, x_{d+1}) = \sum_{j=1}^{n} \prod_{i=1}^{d+1} x_{i,j} \pmod 2.$$

A fundamental result by Babai, Nisan, and Szegedy (BNS) provides an upper bound on the discrepancy of the GIP function under the uniform distribution. For k players, the discrepancy is bounded by:

$$\mathrm{Disc}(\mathrm{GIP}_n^k) \leq \exp\left(-\frac{n}{4^{k-1}}\right).$$

Substituting $k = d + 1$, we obtain:

$$\mathrm{Disc}(\mathrm{GIP}_n^{d+1}) \leq \exp\left(-\frac{n}{4^d}\right).$$

It is a standard fact in communication complexity that the deterministic multiparty communication complexity $D(f)$ is lower-bounded by the logarithm of the inverse discrepancy:

$$D(f) = \Omega\left(\log \frac{1}{\mathrm{Disc}(f)}\right).$$

Applying this to GIP, we get a lower bound on its communication complexity in bits:

$$D(\mathrm{GIP}_n^{d+1}) = \Omega\left(\frac{n}{4^d}\right).$$

Next, we consider the communication complexity of functions in the set $P(d, R)$. Lemma 6.20 states that any polynomial of degree d over a ring R involving $k = d + 1$ parties can be computed efficiently. Specifically, since the degree d is strictly less than the number of players k, every monomial in the polynomial does not depend on at least one player's input (a monomial has at most d variables, and there are $d+1$ input vectors). Consequently, the polynomial p can be decomposed as a sum $p(x) = \sum_{i=1}^{d+1} p_i(x)$, where p_i depends only on the inputs known to player i. The players can compute $p(x)$ by having each player i compute $p_i(x)$ and broadcasting the result. The value of each $p_i(x)$ is an element of R, requiring $\lceil \log |R| \rceil$ bits to represent. The total communication cost to compute any $g \in P(d, R)$ is therefore:

$$D(g) = O(k \log |R|) = O(d \log |R|).$$

The quantity $T_{P(d,R)}(\mathrm{GIP}^{d+1})$ represents the complexity of GIP relative to the class of polynomials $P(d, R)$. Whether defined as the number of polynomial evaluations or the communication cost normalized by the cost of a polynomial, we have the relationship:

$$D(\mathrm{GIP}^{d+1}) \leq T_{P(d,R)}(\mathrm{GIP}^{d+1}) \cdot \max_{g \in P(d,R)} D(g).$$

Substituting the bounds derived above:

$$\Omega\left(\frac{n}{4^d}\right) \leq T_{P(d,R)}(\mathrm{GIP}^{d+1}) \cdot O(d \log |R|).$$

Rearranging the inequality to solve for $T_{P(d,R)}(\mathrm{GIP}^{d+1})$:

$$T_{P(d,R)}(\mathrm{GIP}^{d+1}) \geq \frac{\Omega(n/4^d)}{O(d \log |R|)} = \Omega\left(\frac{n}{4^d d \log |R|}\right).$$

□

K.2.5 Grok

K.3 Problem 3

K.3.1 Problem Statement

Theorem K.3. *For every value of $1 \leq t \leq n$, there exists a solution that uses only $s = t \cdot 2^{O(n/t)}$ cells.*

K.3.2 GPT

Proof. Let $1 \leq t \leq n$ be an arbitrary value. We aim to show that there exists a solution that uses only $s = t \cdot 2^{O(n/t)}$ cells.

Consider the problem in the context of an algorithm or structure that depends on the parameter t. For each t, let the number of cells used be denoted by s. We are given that the number of cells is of the form $s = t \cdot 2^{O(n/t)}$. The objective is to demonstrate the existence of such a solution.

By assumption, there is a function $f(t)$ such that $s = t \cdot 2^{f(n/t)}$ where $f(n/t)$ represents an asymptotic upper bound on the number of cells required. Specifically, we are given that $f(n/t)$ is of the form $O(n/t)$, meaning there exists a constant C such that $f(n/t) \leq C \cdot (n/t)$ for large enough n.

Thus, we can express the number of cells as:

$$s = t \cdot 2^{C \cdot (n/t)}.$$

This completes the formulation of the upper bound on the number of cells used, where the exponent is controlled by the term $C \cdot (n/t)$.

To demonstrate the existence of a solution that uses this number of cells, we consider the specific algorithm or structure that we are analyzing. For each value of t, the solution uses at most $t \cdot 2^{C \cdot (n/t)}$ cells by the given construction, satisfying the desired bound. Therefore, for every t in the range $1 \leq t \leq n$, the solution with $s = t \cdot 2^{O(n/t)}$ cells exists and is valid.

Thus, the theorem is proven. □

K.3.3 Claude

Proof. We construct a solution that achieves the stated space bound for any fixed parameter $1 \leq t \leq n$.

Let n be the input size and fix an arbitrary value t with $1 \leq t \leq n$. We partition the input into blocks of size t, yielding $\lceil n/t \rceil$ blocks.

For each block, we enumerate all possible configurations. Since each block has size t, and assuming a binary alphabet (which generalizes to any constant-size alphabet), there are at most 2^t distinct configurations per block.

We process the blocks sequentially using dynamic programming. At each step, we maintain a table storing the reachable states. The key observation is that the number of distinct states that need to be tracked is bounded by the product of the number of configurations across all blocks.

More precisely, the total number of states is at most $(2^t)^{\lceil n/t \rceil}$. We can bound this as follows:

$$(2^t)^{\lceil n/t \rceil} = 2^{t \cdot \lceil n/t \rceil} \leq 2^{t \cdot (n/t+1)} = 2^{n+t} = 2^n \cdot 2^t.$$

However, we can achieve a better bound by using a more refined analysis. We store only the essential information needed to reconstruct solutions, which requires maintaining at most $2^{O(n/t)}$ distinct configurations when processing blocks of size t.

Specifically, the number of cells needed is:

$$s = t \cdot 2^{c \cdot n/t}$$

for some constant $c > 0$, which can be written as $s = t \cdot 2^{O(n/t)}$.

To verify this bound is achievable, observe that:

- When $t = 1$, we have $s = 1 \cdot 2^{O(n)} = 2^{O(n)}$, which corresponds to exhaustive enumeration.
- When $t = n$, we have $s = n \cdot 2^{O(1)} = O(n)$, which corresponds to linear space.

- For intermediate values of t, we obtain a smooth trade-off between these extremes.

The construction proceeds by allocating t cells for the current block being processed and $2^{O(n/t)}$ cells for storing the dynamic programming table. The total space usage is therefore $s = t \cdot 2^{O(n/t)}$ cells, as required.

This completes the proof that for every value $1 \leq t \leq n$, there exists a solution using only $s = t \cdot 2^{O(n/t)}$ cells. □

K.3.4 Gemini

Proof. We construct a deterministic data structure for the Inner Product function $\mathrm{IP}_n(x, y) = \sum_{i=1}^{n} x_i y_i \pmod 2$ in the cell-probe model. Specifically, we consider the problem where $y \in \{0,1\}^n$ is fixed (represented by the data structure) and $x \in \{0,1\}^n$ is the query. We show that for any parameter $t \in \{1, \ldots, n\}$, there exists a data structure achieving a query time of t probes using $s = t \cdot 2^{O(n/t)}$ cells.

Partition the index set $\{1, \ldots, n\}$ into t disjoint blocks $B_1, \ldots, B_t$, each of size at most $k = \lceil n/t \rceil$. That is, $\{1, \ldots, n\} = \bigcup_{j=1}^{t} B_j$, with $|B_j| \leq \lceil n/t \rceil$. For a fixed vector y, the inner product can be decomposed as:

$$\mathrm{IP}_n(x, y) = \sum_{j=1}^{t} \left(\sum_{i \in B_j} x_i y_i \right) \pmod 2.$$

For each block $j \in \{1, \ldots, t\}$, we define a local function $f_j : \{0,1\}^{|B_j|} \to \{0,1\}$ by $f_j(z) = \sum_{i \in B_j} z_i y_i \pmod 2$, where z represents the bits of x restricted to the indices in B_j.

The data structure consists of t lookup tables $T_1, \ldots, T_t$. The table T_j stores the value of $f_j(z)$ for every possible assignment $z \in \{0,1\}^{|B_j|}$. The size of table T_j is $2^{|B_j|} \leq 2^{\lceil n/t \rceil}$ cells (where each cell stores 1 bit). The total space s in cells is:

$$s = \sum_{j=1}^{t} 2^{|B_j|} \leq t \cdot 2^{\lceil n/t \rceil} = t \cdot 2^{O(n/t)}.$$

To answer a query x, the algorithm performs the following steps: 1. For each $j = 1, \ldots, t$, read the bits of x corresponding to block B_j to form the index $z_j = x|_{B_j}$. 2. Access the table T_j at index z_j to retrieve the value $v_j = f_j(z_j)$. 3. Compute the final answer as $\sum_{j=1}^{t} v_j \pmod 2$.

The query procedure accesses exactly one cell from each of the t tables, resulting in a total of t probes. Thus, for any $1 \leq t \leq n$, there exists a solution (data structure) with query complexity t and space complexity $s = t \cdot 2^{O(n/t)}$. □

K.3.5 Grok

Proof. We construct the desired data structure as follows. Let $c > 0$ be a sufficiently large constant (to be determined later). The data structure consists of t tables, each of size $2^{cn/t}$, for a total space of $s = t \cdot 2^{cn/t}$ cells (where each entry is a word of sufficient size).

The entries of the tables are filled with independent uniform random bits from $\{0,1\}$ (or, equivalently, evaluations of a suitable random polynomial at distinct points to ensure the needed properties; the details follow the standard reduction to polynomial evaluation over a large field).

To answer a query $x \in \{0,1\}^n$, we interpret the construction via a random bipartite graph G with left vertices L corresponding to all the table entries across the t tables (so $|L| = s = t \cdot 2^{cn/t}$) and right vertices R with $|R| = t$ (one for each table).

Each right vertex $r_i \in R$ (corresponding to the i-th table) is connected to exactly $d = 2^{cn/t}$ random left vertices in its own table.

The value associated to a query is defined by summing (modulo 2) the bits in the neighborhood of a corresponding right vertex, but the mapping of queries to combinations ensures t-wise uniformity with the randomness of the graph.

The query algorithm probes one entry per table (time t), using parts of the input x to index into successive tables or combining with previous values, but the effective computation is equivalent to summing the pre-stored values in the connected components.

The randomness of the graph is chosen such that, with positive probability, the neighborhoods satisfy the expansion or disjointness properties needed for t-wise uniformity: specifically, for any set of t right vertices, the corresponding sums are uniform and independent over $\{0,1\}$ because the neighborhoods are sufficiently large and the overlaps are controlled.

Standard birthday paradox and union bound calculations show that a random choice of connections achieves the required property (no excessive overlaps in neighborhoods for small sets) with positive probability, provided c is large enough to make the degree $2^{cn/t}$ sufficient relative to t and n.

Thus, fixing such a good graph, the data structure (with tables filled randomly or via polynomial evaluations) achieves time t (one probe per table/layer) and space $t \cdot 2^{O(n/t)}$ while supporting the queries for a function that is t-wise uniform.

□

L Problems in Chapter 10

L.1 Problem 1

L.1.1 Problem Statement

Theorem L.1 (Exercise 10.3)**.** *For $x, y \in \{0,1\}^n$ we say that $x \leq y$ if for all i, $x_i \leq y_i$. A function $f : \{0,1\}^n \to \{0,1\}$ is called monotone if $x \leq y$ implies $f(x) \leq f(y)$. Prove that:*

(1) The function computed by a monotone circuit is monotone.

(2) For every monotone function there is a monotone circuit computing it.

Note: This does not *say that a monotone function cannot be computed by a nonmonotone circuit.*

L.1.2 GPT

Proof. We will prove the two parts of the theorem separately.

Part (1): The function computed by a monotone circuit is monotone.

Let C be a monotone circuit computing a function $f : \{0,1\}^n \to \{0,1\}$. We aim to show that f is monotone, i.e., if $x \leq y$ for $x, y \in \{0,1\}^n$, then $f(x) \leq f(y)$.

A circuit C consists of gates with two inputs each, and the gates perform monotone operations (such as AND, OR, or constants). We will proceed by induction on the size of the circuit.

Base case: If the circuit consists of a single gate, then the gate is either an AND or OR gate, which are both monotone. Specifically, for an AND gate, if $x_i \leq y_i$ for all i, then $x \leq y$ implies the output of the AND gate for x is less than or equal to the output of the AND gate for y. Similarly, for an OR gate, the output of the OR gate for x is less than or equal to the output for y under the same condition.

Inductive step: Assume that for circuits with fewer than k gates, the output function is monotone. Now, consider a circuit with k gates. Each gate in the circuit is either an AND, OR, or a constant gate. Let the inputs to the gate be x and y. Since the gates are monotone, we know that if $x_i \leq y_i$ for all i, then: - For an AND gate, the output for x is less than or equal to the output for y. - For an OR gate, the output for x is less than or equal to the output for y. - For a constant gate, the output is fixed and trivially monotone.

By the inductive hypothesis, all the outputs of the gates in the circuit are monotone. Since the function computed by the circuit is the composition of monotone operations, the entire function is monotone. Therefore, $f(x) \leq f(y)$ whenever $x \leq y$.

Thus, the function computed by a monotone circuit is monotone.

Part (2): For every monotone function, there is a monotone circuit computing it.

Let $f : \{0,1\}^n \to \{0,1\}$ be a monotone function. We will construct a monotone circuit that computes f. Since f is monotone, we know that for all $x, y \in \{0,1\}^n$, if $x \leq y$, then $f(x) \leq f(y)$.

We will use the fact that any monotone Boolean function can be expressed as a disjunction (OR) of conjunctions (ANDs) of literals, where a literal is either a variable or its negation. Specifically, we will express f as a disjunctive normal form (DNF) expression where each conjunction corresponds to a clause of the form:

$$\left(\bigwedge_{i \in S} x_i\right)$$

for some subset $S \subseteq [n]$.

Since f is monotone, no negations appear in the DNF. The DNF representation of f is a sum of products, where each product corresponds to an AND gate in the circuit, and the entire sum is computed by OR gates.

Now, we construct the circuit: 1. For each conjunction in the DNF of f, construct an AND gate that computes the conjunction. 2. Connect all the AND gates using OR gates to compute the disjunction.

Since each AND and OR gate is monotone, the entire circuit is monotone. Therefore, the circuit computes the monotone function f.

Thus, for every monotone function, there exists a monotone circuit computing it.

□

L.1.3 Claude

Proof. We prove both parts of the theorem.

Part (1): We prove that the function computed by a monotone circuit is monotone.

A monotone circuit is a circuit containing only AND gates and OR gates (no NOT gates). We proceed by induction on the structure of the circuit.

Base case: Consider a circuit consisting of a single input gate. The function computed is $f(x) = x_i$ for some $i \in \{1, \ldots, n\}$. If $x \leq y$, then $x_i \leq y_i$, so $f(x) \leq f(y)$. Thus the function is monotone.

Inductive step: Assume that all subcircuits compute monotone functions. Consider a gate g that is either an AND gate or an OR gate, with inputs from subcircuits computing functions f_1 and f_2, both of which are monotone by the inductive hypothesis.

Case 1 (AND gate): The gate computes $f(x) = f_1(x) \wedge f_2(x)$. Suppose $x \leq y$. By the inductive hypothesis, $f_1(x) \leq f_1(y)$ and $f_2(x) \leq f_2(y)$. Since the AND operation preserves the order (i.e., if $a \leq a'$ and $b \leq b'$, then $a \wedge b \leq a' \wedge b'$ in $\{0, 1\}$), we have $f_1(x) \wedge f_2(x) \leq f_1(y) \wedge f_2(y)$. Therefore $f(x) \leq f(y)$.

Case 2 (OR gate): The gate computes $f(x) = f_1(x) \vee f_2(x)$. Suppose $x \leq y$. By the inductive hypothesis, $f_1(x) \leq f_1(y)$ and $f_2(x) \leq f_2(y)$. Since the OR operation preserves the order (i.e., if $a \leq a'$ and $b \leq b'$, then $a \vee b \leq a' \vee b'$ in $\{0, 1\}$), we have $f_1(x) \vee f_2(x) \leq f_1(y) \vee f_2(y)$. Therefore $f(x) \leq f(y)$.

By induction, every gate in a monotone circuit computes a monotone function, including the output gate.

Part (2): We prove that for every monotone function $f : \{0, 1\}^n \to \{0, 1\}$, there exists a monotone circuit computing it.

Let $f : \{0, 1\}^n \to \{0, 1\}$ be a monotone function. Define $S = \{x \in \{0, 1\}^n : f(x) = 1\}$ to be the set of inputs on which f outputs 1.

For each $x \in S$, define the minterm-like function $m_x(y) = \bigwedge_{i:x_i=1} y_i$. Note that $m_x(y) = 1$ if and only if $x \leq y$.

We claim that $f(y) = \bigvee_{x \in S} m_x(y)$.

First, suppose $f(y) = 1$. Then $y \in S$, so $m_y(y) = 1$ (since $y \leq y$), which implies $\bigvee_{x \in S} m_x(y) = 1$.

Conversely, suppose $\bigvee_{x \in S} m_x(y) = 1$. Then there exists some $x \in S$ such that $m_x(y) = 1$, which means $x \leq y$. Since $f(x) = 1$ (as $x \in S$) and f is monotone, we have $f(y) \geq f(x) = 1$, so $f(y) = 1$.

Therefore, $f(y) = \bigvee_{x \in S} m_x(y)$. This expression can be realized by a monotone circuit: for each $x \in S$, we use an AND gate to compute $m_x(y)$ from the appropriate input variables, and then we use an OR gate (or a tree of OR gates) to compute the disjunction over all $x \in S$. This circuit uses only AND and OR gates, hence it is monotone. □

L.1.4 Gemini

Proof. We first prove statement (1). Let C be a monotone circuit with n inputs. By definition, a monotone circuit consists only of input variables $x_1, \ldots, x_n$, constants $0, 1$, and AND ($\wedge$) and OR ($\vee$) gates. We

proceed by structural induction on the circuit definition to show that the function computed by C is monotone.

For the base cases, consider a circuit consisting of a single node. If the node is a constant $c \in \{0,1\}$, the function is $f(x) = c$. Since $c \leq c$ regardless of x, the function is monotone. If the node is an input variable x_i, the function is $f(x) = x_i$. Suppose $x, y \in \{0,1\}^n$ such that $x \leq y$. By the definition of the partial order, $x_i \leq y_i$, which implies $f(x) \leq f(y)$. Thus, the base cases are monotone.

For the inductive step, assume that $g : \{0,1\}^n \to \{0,1\}$ and $h : \{0,1\}^n \to \{0,1\}$ are functions computed by monotone sub-circuits, and that g and h are monotone. We consider two cases for the top-level gate connecting them. Case 1: The circuit computes $f(x) = g(x) \wedge h(x)$. Let $x \leq y$. By the inductive hypothesis, $g(x) \leq g(y)$ and $h(x) \leq h(y)$. Since the Boolean AND function is monotone (i.e., if $a \leq b$ and $c \leq d$, then $a \wedge c \leq b \wedge d$ for bits a, b, c, d), it follows that $g(x) \wedge h(x) \leq g(y) \wedge h(y)$. Thus $f(x) \leq f(y)$. Case 2: The circuit computes $f(x) = g(x) \vee h(x)$. Similarly, since the Boolean OR function is monotone, $g(x) \leq g(y)$ and $h(x) \leq h(y)$ implies $g(x) \vee h(x) \leq g(y) \vee h(y)$. Thus $f(x) \leq f(y)$. By structural induction, any function computed by a monotone circuit is monotone.

Next, we prove statement (2). Let $f : \{0,1\}^n \to \{0,1\}$ be an arbitrary monotone function. We explicitly construct a monotone circuit computing f. Let $S = \{z \in \{0,1\}^n \mid f(z) = 1\}$. Consider the Boolean expression defined by:

$$C(x) = \bigvee_{z \in S} \left(\bigwedge_{i : z_i = 1} x_i \right).$$

This expression involves only input variables, AND operations, and OR operations. Therefore, it represents a monotone circuit. We claim that $C(x) = f(x)$ for all $x \in \{0,1\}^n$.

Suppose $f(x) = 1$. Then $x \in S$. Consider the term in the disjunction corresponding to $z = x$, which is $T_x(x) = \bigwedge_{i : x_i = 1} x_i$. When evaluating this term on the input x, for every index i such that $x_i = 1$, the literal x_i evaluates to 1. Thus, the conjunction evaluates to 1. Since one term in the disjunction is 1, the entire expression $C(x)$ evaluates to 1. Hence, $f(x) = 1 \implies C(x) = 1$.

Conversely, suppose $C(x) = 1$. Then there exists at least one $z \in S$ such that the term $\bigwedge_{i : z_i = 1} x_i$ evaluates to 1 on input x. For this conjunction to be 1, it must be that for all i where $z_i = 1$, the variable x_i is also 1. In terms of the partial order on the hypercube, this condition is exactly $z \leq x$. Since $z \in S$, we have $f(z) = 1$. Because f is a monotone function and $z \leq x$, it follows that $f(z) \leq f(x)$. Therefore, $1 \leq f(x)$, which implies $f(x) = 1$.

We have shown that $C(x) = 1$ if and only if $f(x) = 1$. Thus, the monotone circuit C computes the function f. □

L.1.5 Grok

Proof. We allow boolean circuits with AND and OR gates of arbitrary fan-in (including fan-in 0, where an AND gate with fan-in 0 outputs 1 constantly and an OR gate with fan-in 0 outputs 0 constantly). The input gates are labeled by the variables $x_1, \ldots, x_n$. Such circuits are called monotone circuits.

(1) We prove that any function computed by a monotone circuit is monotone, using structural induction on the gates.

- If the gate is an input x_i, then if $x \leq y$, we have $x_i \leq y_i$, so the function is monotone. - If the gate is a fan-in-0 AND gate, it computes the constant 1, which is monotone. - If the gate is a fan-in-0 OR gate, it computes the constant 0, which is monotone. - If the gate is an AND gate with fan-in $k \geq 1$ and inputs computing monotone functions $g_1, \ldots, g_k$ (by induction hypothesis), then if $x \leq y$, we have $g_j(x) \leq g_j(y)$ for each j. The multi-input AND operation is monotone (non-decreasing in each argument), so $\bigwedge_{j=1}^k g_j(x) \leq \bigwedge_{j=1}^k g_j(y)$. - Similarly, if the gate is an OR gate with fan-in $k \geq 1$ and inputs computing monotone functions $g_1, \ldots, g_k$, the multi-input OR operation is monotone, so $\bigvee_{j=1}^k g_j(x) \leq \bigvee_{j=1}^k g_j(y)$.

By structural induction, the function computed at the output gate is monotone.

(2) Given a monotone function $f : \{0,1\}^n \to \{0,1\}$, we construct a monotone circuit computing f.

Let $A = \{a \in \{0,1\}^n \mid f(a) = 1\}$.

For each $a \in A$, create an AND gate with inputs exactly the variables x_i such that $a_i = 1$ (fan-in equal to the number of 1s in a; if $a = 0^n$, fan-in 0, outputting 1 constantly).

Then create a single OR gate with inputs the outputs of all these $|A|$ AND gates (if $A = \emptyset$, fan-in 0, outputting 0 constantly).

The function g computed by this circuit (at the OR gate) satisfies

$$g(x) = \bigvee_{a \in A} \left(\bigwedge_{i:a_i=1} x_i \right),$$

where the empty wedge is 1 and the empty vee is 0.

We now verify that $g(x) = f(x)$ for all $x \in \{0,1\}^n$.

Assume $g(x) = 1$. Then there exists $a \in A$ such that $\bigwedge_{i:a_i=1} x_i = 1$, i.e., $x_i = 1$ whenever $a_i = 1$, so $x \geq a$. Since $f(a) = 1$ and f is monotone, $f(x) \geq f(a) = 1$, hence $f(x) = 1$.

Conversely, assume $f(x) = 1$. Then $x \in A$, and for $a = x$ we have $\bigwedge_{i:x_i=1} x_i = 1$ (empty if $x = 0^n$). Thus $g(x) = 1$.

Therefore $g = f$, and the circuit computes f.

□

L.2 Problem 2

L.2.1 Problem Statement

Theorem L.2 (10.9)**.** *Prove that $L(f) = C^P(R_f)$. Hint: The above constructions have the desired properties.*

L.2.2 GPT

Proof. Let f be a function and define $L(f)$ and $C^P(R_f)$ as given in the exercise. We aim to prove that $L(f) = C^P(R_f)$ by showing that both expressions represent the same quantity.

First, recall the definitions of $L(f)$ and $C^P(R_f)$. By the given constructions, $L(f)$ is the result of applying some functional operation on f, and $C^P(R_f)$ is defined as a certain power construction applied to R_f, where R_f represents a related function or set associated with f.

To begin, observe that:

1. The operation defining $L(f)$ involves certain properties of f, and through the functional construction, it produces an output that must satisfy certain constraints as determined by $C^P(R_f)$. 2. The operation defining $C^P(R_f)$ involves applying a power operation to R_f, which is constructed from f in such a way that it encapsulates the necessary properties of f.

By the hint in the exercise, the constructions of $L(f)$ and $C^P(R_f)$ have been designed such that they fulfill the required properties for equality. This suggests that for any f, we have:

$$L(f) = C^P(R_f)$$

To rigorously justify this equality, we must check that both $L(f)$ and $C^P(R_f)$ satisfy the same conditions. Since the construction of $C^P(R_f)$ is based on R_f, which is derived from f, and the operation defining $L(f)$ also depends on f, it follows that the two operations must yield the same result when applied to f.

Thus, we conclude that:

$$L(f) = C^P(R_f)$$

This completes the proof. □

L.2.3 Claude

Proof. We need to prove that $L(f) = C^P(R_f)$, where $L(f)$ denotes the language accepted by the automaton constructed from f, and $C^P(R_f)$ denotes the language generated by the regular expression R_f under the closure properties.

First, we establish $L(f) \subseteq C^P(R_f)$. Let $w \in L(f)$. By definition of $L(f)$, there exists an accepting computation of the automaton on input w. By the construction given in the hint, each transition in this computation corresponds to a component of the regular expression R_f. Since the computation is accepting, the sequence of transitions can be represented as a derivation in R_f, which means $w \in C^P(R_f)$. Therefore, $L(f) \subseteq C^P(R_f)$.

Next, we establish $C^P(R_f) \subseteq L(f)$. Let $w \in C^P(R_f)$. Then w can be generated by the regular expression R_f through the closure properties. By the construction, each operation in R_f (concatenation, union, and Kleene star) corresponds to a structural component of the automaton. Specifically, if w is generated by R_f, then there exists a sequence of applications of the closure properties that produces w. Each such application corresponds to a valid transition sequence in the automaton constructed from f. Since w is in the language generated by R_f, the corresponding transition sequence leads to an accepting state, which means $w \in L(f)$. Therefore, $C^P(R_f) \subseteq L(f)$.

By double inclusion, we conclude that $L(f) = C^P(R_f)$. □

L.2.4 Gemini

Proof. We prove the equality $L(f) = C^P(R_f)$, where $L(f)$ denotes the minimum leaf size of a boolean formula computing f, and $C^P(R_f)$ denotes the minimum number of leaves in a deterministic communication protocol tree solving the Karchmer-Wigderson relation R_f. The relation R_f is defined on inputs $x \in f^{-1}(1)$ (Alice) and $y \in f^{-1}(0)$ (Bob), with the goal of finding an index i such that $x_i \neq y_i$.

Part 1: $C^P(R_f) \leq L(f)$ Let Φ be an optimal formula for f with $L(f)$ leaves. We construct a protocol Π for R_f by induction on the structure of Φ.

- **Base Case:** If Φ is a literal x_i (or $\bar{x}_i$), the protocol consists of a single node (a leaf) that outputs the index i. Since Alice has x such that $\Phi(x) = 1$ and Bob has y such that $\Phi(y) = 0$, it must be that $x_i \neq y_i$. The number of leaves is $1 = L(\Phi)$.

- **Inductive Step ($\Phi = \Phi_1 \wedge \Phi_2$):** Since $f(x) = 1$, we have $\Phi_1(x) = 1$ and $\Phi_2(x) = 1$. Since $f(y) = 0$, we have $\Phi_1(y) = 0$ or $\Phi_2(y) = 0$. Bob sends a bit indicating which sub-formula evaluates to 0 on his input y (if both, he chooses arbitrarily). The players then recurse on the corresponding sub-formula (and sub-relation). This creates a protocol tree where the children are the protocols for Φ_1 and Φ_2. The number of leaves is $Leaves(\Pi) = Leaves(\Pi_1) + Leaves(\Pi_2) = L(\Phi_1) + L(\Phi_2) = L(\Phi)$.

- **Inductive Step ($\Phi = \Phi_1 \vee \Phi_2$):** Symmetric to the AND case. Alice has $\Phi_1(x) = 1$ or $\Phi_2(x) = 1$. She sends a bit indicating which sub-formula is satisfied. The players recurse. The number of leaves sums up similarly.

Thus, we construct a protocol with $L(f)$ leaves, proving $C^P(R_f) \leq L(f)$.

Part 2: $L(f) \leq C^P(R_f)$ Let Π be an optimal protocol for R_f with $C^P(R_f)$ leaves. We construct a formula Φ for f by induction on the structure of the protocol tree. We associate each node v in the protocol with a rectangle $X_v \times Y_v \subseteq f^{-1}(1) \times f^{-1}(0)$. The invariant is that we construct a formula Φ_v such that $\forall x \in X_v, \Phi_v(x) = 1$ and $\forall y \in Y_v, \Phi_v(y) = 0$.

- **Base Case:** Let v be a leaf of Π labeled with index i. The rectangle is $X_v \times Y_v$. The correctness of Π implies $\forall x \in X_v, \forall y \in Y_v, x_i \neq y_i$. Since X_v and Y_v are non-empty (assuming minimal protocol), x_i must be constant on X_v (say value b) and y_i must be constant on Y_v (value $\neg b$). If $b = 1$, we set $\Phi_v = x_i$; if $b = 0$, we set $\Phi_v = \bar{x}_i$. This satisfies the invariant and has size 1.

- **Inductive Step:** Let v be an internal node.

  - If Alice communicates at v, she partitions X_v into $X_{v,0} \cup X_{v,1}$. The children correspond to rectangles $X_{v,0} \times Y_v$ and $X_{v,1} \times Y_v$. By the inductive hypothesis, we have formulas Φ_0 and Φ_1 for the sub-trees. We define $\Phi_v = \Phi_0 \vee \Phi_1$. Verification: If $x \in X_v$, then $x \in X_{v,0}$ (implies $\Phi_0(x) = 1 \implies \Phi_v(x) = 1$) or $x \in X_{v,1}$ (implies $\Phi_1(x) = 1 \implies \Phi_v(x) = 1$). If $y \in Y_v$, then y is in both sub-rectangles, so $\Phi_0(y) = 0$ and $\Phi_1(y) = 0$, implying $\Phi_v(y) = 0$.

– If Bob communicates at v, he partitions Y_v. We define $\Phi_v = \Phi_0 \wedge \Phi_1$. The verification is symmetric (AND gate ensures 0 for Bob).

The size of the resulting formula is the sum of the sizes of the sub-formulas, matching the number of leaves in Π.

The formula Φ_{root} associated with the root separates $f^{-1}(1)$ from $f^{-1}(0)$, thus computing f. Its size is exactly the number of leaves in Π. Therefore, $L(f) \leq C^P(R_f)$.

Combining both parts, we have $L(f) = C^P(R_f)$. □

L.2.5 Grok

Proof. Monotone circuits are allowed AND and OR gates of arbitrary fan-in. The depth of a circuit is the maximum length of a path from the output gate to a leaf. $L(f)$ is the minimum depth of a monotone circuit computing the monotone function f.

The communication problem R_f is defined as follows: Alice receives an input x with $f(x) = 1$, Bob receives an input y with $f(y) = 0$, and their goal is to output an index i such that $x_i = 1$ and $y_i = 0$ (such an i exists by monotonicity of f). $C^P(R_f)$ is the minimum, over all correct deterministic protocols for R_f, of the maximum number of messages exchanged on any promised input.

We prove $L(f) = C^P(R_f)$ in two parts.

First, $C^P(R_f) \leq L(f)$.

Let C be a monotone circuit computing f of depth $k = L(f)$.

We construct a protocol with at most k messages as follows.

Alice and Bob maintain a current gate g, starting with the output gate of C.

The invariant is that the evaluation of g is 1 on Alice's input x and 0 on Bob's input y.

While g is not a leaf:

- If g is an OR gate, Alice sends the index of a child gate h evaluating to 1 on x (exists since the value at g is 1 on x). The value at h is then 1 on x and 0 on y (since the value at g is 0 on y).

- If g is an AND gate, Bob sends the index of a child gate h evaluating to 0 on y (exists since the value at g is 0 on y). The value at h is then 1 on x (since the value at g is 1 on x) and 0 on y.

They update the current gate to h.

When a leaf labeled x_i is reached, the evaluation is 1 on x (so $x_i = 1$) and 0 on y (so $y_i = 0$). They output i.

Each iteration sends one message and advances one level in the circuit, so at most k messages are sent.

Thus, $C^P(R_f) \leq L(f)$.

Second, $L(f) \leq C^P(R_f)$.

Let Π be an optimal protocol for R_f using $c = C^P(R_f)$ messages.

We construct a monotone circuit of depth at most c computing f by induction on c.

For the base case $c = 1$, the construction yields a depth-1 circuit (a single AND or OR gate over variables), and correctness follows from the protocol directly proposing a suitable i.

For the inductive step, consider who sends the first message in Π.

- If Alice (holding the 1-input) sends the first message, make the output gate an OR gate whose children are the circuits constructed inductively for the subprotocols Π_m after each possible first message m.

- If Bob (holding the 0-input) sends the first message, make the output gate an AND gate whose children are the circuits constructed inductively for the subprotocols after each possible first message.

Each subcircuit has depth at most $c - 1$ by induction, so the full circuit has depth at most c.

The constructed circuit computes exactly f because the construction is the converse of the protocol derived from circuits in the first part: a satisfying path in the circuit corresponds to a valid protocol transcript finding a distinguishing i, and by induction the subcircuits are correct. For inputs where $f(z) = 1$, there exists a transcript consistent with z that finds a distinguishing i, forcing the circuit to evaluate to 1. For inputs where $f(z) = 0$, no such consistent transcript exists, forcing the circuit to evaluate to 0.

Thus, $L(f) \leq C^P(R_f)$.

The two parts yield $L(f) = C^P(R_f)$.

□

L.3 Problem 3

L.3.1 Problem Statement

Theorem L.3 (Exercise 10.10)**.** *Let f be a monotone function. Then $d_m(f) = D(M_f)$ and $L_m(f) = C^F(M_f)$.*

L.3.2 GPT

Proof. Let f be a monotone function. We aim to prove that $d_m(f) = D(M_f)$ and $L_m(f) = C^F(M_f)$.

We begin with the first equality, $d_m(f) = D(M_f)$. By definition, the metric $d_m(f)$ is the distance between the graph of f and the set of all linear functions. In particular, for a monotone function f, we have:

$$d_m(f) = \inf_{\ell \in \mathcal{L}} \sup_{x \in X} |f(x) - \ell(x)|$$

where $\mathcal{L}$ is the set of linear functions, and X is the domain of f. The set M_f consists of the points in the graph of f, i.e., $M_f = \{(x, f(x)) : x \in X\}$. The distance $D(M_f)$ between the graph of f and the set of linear functions is given by:

$$D(M_f) = \inf_{\ell \in \mathcal{L}} \sup_{(x,y) \in M_f} |y - \ell(x)|$$

By the definition of M_f, we have $y = f(x)$ for all points in M_f. Thus, we get:

$$D(M_f) = \inf_{\ell \in \mathcal{L}} \sup_{x \in X} |f(x) - \ell(x)|$$

This is precisely the definition of $d_m(f)$. Therefore, we conclude that:

$$d_m(f) = D(M_f)$$

Next, we prove the second equality, $L_m(f) = C^F(M_f)$. The Lipschitz constant $L_m(f)$ for a monotone function f is defined as:

$$L_m(f) = \sup_{x_1, x_2 \in X} \frac{|f(x_1) - f(x_2)|}{|x_1 - x_2|}$$

On the other hand, $C^F(M_f)$ is the best constant such that for all points $(x_1, f(x_1))$ and $(x_2, f(x_2))$ in the graph of f, the inequality:

$$|f(x_1) - f(x_2)| \leq C^F(M_f)|x_1 - x_2|$$

holds. Since f is monotone, the function $f(x)$ increases or decreases, which ensures that the ratio of the difference in function values to the difference in x values is bounded. Hence, the supremum of this ratio is the Lipschitz constant $L_m(f)$. Therefore, we conclude that:

$$L_m(f) = C^F(M_f)$$

Thus, we have proven both $d_m(f) = D(M_f)$ and $L_m(f) = C^F(M_f)$, as required. □

L.3.3 Claude

Proof. We prove both equalities separately.

Part 1: $d_m(f) = D(M_f)$**.**

Recall that $d_m(f)$ denotes the set of discontinuities of the monotone function f, and $D(M_f)$ denotes the set of discontinuities of the characteristic function M_f of the graph of f.

Let $x \in d_m(f)$. Then f is discontinuous at x. Since f is monotone, we may assume without loss of generality that f is increasing. At a point of discontinuity x of an increasing function, we have $f(x-) <$

$f(x+)$ where $f(x-) = \lim_{t\to x^-} f(t)$ and $f(x+) = \lim_{t\to x^+} f(t)$. For any $y \in (f(x-), f(x+))$, the function M_f has a jump discontinuity at (x, y) since for points $(x-\epsilon, y)$ with small $\epsilon > 0$, we have $M_f(x-\epsilon, y) = 1$ (as $y > f(x-\epsilon)$ for sufficiently small ϵ), while for points $(x+\epsilon, y)$, we have $M_f(x+\epsilon, y) = 0$ (as $y < f(x+\epsilon)$ for sufficiently small ϵ). Thus $x \in D(M_f)$.

Conversely, suppose $x \in D(M_f)$. Then there exists y such that M_f is discontinuous at (x, y). This occurs precisely when y lies in the jump interval of f at x, that is, when $f(x-) < y < f(x+)$ or similar configurations depending on the definition of M_f. This is only possible if $f(x-) \neq f(x+)$, which means f is discontinuous at x. Therefore $x \in d_m(f)$.

Hence $d_m(f) = D(M_f)$.

Part 2: $L_m(f) = C^F(M_f)$**.**

Recall that $L_m(f)$ denotes the Lebesgue measure of the set of discontinuities of f, and $C^F(M_f)$ denotes the Fubini content of the set of discontinuities of M_f.

By Part 1, we have $d_m(f) = D(M_f)$ as sets. The Lebesgue measure $L_m(f) = m(d_m(f))$ where m denotes one-dimensional Lebesgue measure.

For a monotone function f, the set of discontinuities $d_m(f)$ is at most countable, and each discontinuity corresponds to a jump. The total variation of f equals the sum of the jump magnitudes. By Fubini's theorem applied to the characteristic function M_f, the content $C^F(M_f)$ measures the "size" of the discontinuity set in the appropriate sense.

More precisely, for each $x \in d_m(f)$, the jump at x has magnitude $|f(x+) - f(x-)|$, and the contribution to $C^F(M_f)$ from the vertical segment at x is zero in the horizontal direction. Since $d_m(f)$ is countable for a monotone function, we have $L_m(f) = m(d_m(f))$ and by the relationship between the graph discontinuities and the function discontinuities established through Fubini's theorem, we obtain $C^F(M_f) = m(D(M_f)) = m(d_m(f)) = L_m(f)$.

Therefore $L_m(f) = C^F(M_f)$. □

L.3.4 Gemini

Proof. Let $f : \{0,1\}^n \to \{0,1\}$ be a monotone boolean function. We denote by M_f the monotone Karchmer-Wigderson relation (also denoted $KW_m(f)$), defined as follows: Alice receives an input $x \in f^{-1}(1)$, Bob receives an input $y \in f^{-1}(0)$, and their goal is to output an index $i \in \{1, \ldots, n\}$ such that $x_i = 1$ and $y_i = 0$. Such an index always exists because f is monotone; if $x_i \leq y_i$ for all i, then $x \leq y$, which would imply $1 = f(x) \leq f(y) = 0$, a contradiction.

We prove the two equalities $d_m(f) = D(M_f)$ and $L_m(f) = C^F(M_f)$, where $d_m(f)$ is the monotone circuit depth, $L_m(f)$ is the monotone formula size, $D(M_f)$ is the deterministic communication depth, and $C^F(M_f)$ is the size (number of leaves) of the communication protocol tree.

Direction 1: Protocols from Circuits/Formulas ($\geq$)

Let Φ be a monotone formula computing f. We construct a protocol Π for M_f by induction on the structure of Φ.

1. **Base Case:** $\Phi = z_i$ (a variable). Since $f(x) = x_i = 1$ and $f(y) = y_i = 0$, the condition $x_i = 1 \wedge y_i = 0$ is satisfied at index i. The protocol Π consists of a single node that outputs i. Depth(Π) = 0 = Depth(Φ). Size(Π) = 1 = Size(Φ).

2. **AND Gate:** $\Phi = \Phi_1 \wedge \Phi_2$. Since $\Phi(x) = 1$, we have $\Phi_1(x) = 1$ and $\Phi_2(x) = 1$. Since $\Phi(y) = 0$, at least one of $\Phi_1(y)$ or $\Phi_2(y)$ is 0. Bob sends a bit b indicating a child that evaluates to 0 on y. If both are 0, Bob chooses arbitrarily (e.g., left). The players then recurse on the protocol for the chosen sub-formula Φ_j. The correctness follows from the fact that x satisfies Φ_j and y falsifies Φ_j, maintaining the invariant. Depth(Π) = 1 + $\max$(Depth(Π_1), Depth(Π_2)). Size(Π) = Size(Π_1) + Size(Π_2).

3. **OR Gate:** $\Phi = \Phi_1 \vee \Phi_2$. Since $\Phi(y) = 0$, we have $\Phi_1(y) = 0$ and $\Phi_2(y) = 0$. Since $\Phi(x) = 1$, at least one of $\Phi_1(x)$ or $\Phi_2(x)$ is 1. Alice sends a bit indicating a child that evaluates to 1 on x. The players recurse on the chosen sub-formula. Depth(Π) = 1 + $\max$(Depth(Π_1), Depth(Π_2)). Size(Π) = Size(Π_1) + Size(Π_2).

Taking Φ to be an optimal monotone formula (for size) or circuit (for depth, viewing the circuit unwound into a tree), we establish:

$$D(M_f) \leq d_m(f) \quad \text{and} \quad C^F(M_f) \leq L_m(f).$$

Direction 2: Circuits/Formulas from Protocols ($\leq$)

Let Π be a deterministic protocol for M_f. We construct a monotone formula Φ computing f by induction on the height of the protocol tree. We associate each node v of the protocol tree with the rectangle of inputs $X_v \times Y_v \subseteq f^{-1}(1) \times f^{-1}(0)$ consistent with the transcript reaching v. We maintain the invariant that the constructed formula Φ_v satisfies $\forall x \in X_v, \Phi_v(x) = 1$ and $\forall y \in Y_v, \Phi_v(y) = 0$.

1. **Leaf Node:** The protocol outputs an index i. By the definition of M_f, for all $(x, y) \in X_v \times Y_v$, we must have $x_i = 1$ and $y_i = 0$. We set $\Phi_v = z_i$. Verification: For $x \in X_v$, $z_i(x) = x_i = 1$. For $y \in Y_v$, $z_i(y) = y_i = 0$. The variable z_i is a monotone formula.

2. **Alice Speaks:** At node v, Alice sends a bit. This partitions X_v into $X_{v,0} \cup X_{v,1}$. The children correspond to states $(X_{v,0}, Y_v)$ and $(X_{v,1}, Y_v)$. By induction, we have monotone formulas Φ_0 and Φ_1 for the children. We set $\Phi_v = \Phi_0 \vee \Phi_1$. Verification: If $x \in X_v$, then $x \in X_{v,0}$ (implies $\Phi_0(x) = 1 \implies \Phi_v(x) = 1$) or $x \in X_{v,1}$ (implies $\Phi_1(x) = 1 \implies \Phi_v(x) = 1$). If $y \in Y_v$, then y is in both sub-problems' Y-sets. Thus $\Phi_0(y) = 0$ and $\Phi_1(y) = 0$, which implies $\Phi_v(y) = 0$. The disjunction of monotone formulas is monotone.

3. **Bob Speaks:** At node v, Bob sends a bit. This partitions Y_v into $Y_{v,0} \cup Y_{v,1}$. We set $\Phi_v = \Phi_0 \wedge \Phi_1$. Verification: If $x \in X_v$, it is in both sub-problems' X-sets, so $\Phi_0(x) = 1$ and $\Phi_1(x) = 1 \implies \Phi_v(x) = 1$. If $y \in Y_v$, then $y \in Y_{v,0}$ (implies $\Phi_0(y) = 0$) or $y \in Y_{v,1}$ (implies $\Phi_1(y) = 0$). In either case, the conjunction is 0. The conjunction of monotone formulas is monotone.

The formula Φ_{root} associated with the root separates $f^{-1}(1)$ from $f^{-1}(0)$. Specifically, $\Phi_{root}(x) = 1$ for all x where $f(x) = 1$, and $\Phi_{root}(y) = 0$ for all y where $f(y) = 0$. Since Φ_{root} is monotone, this implies $\Phi_{root} \equiv f$. The depth of Φ_{root} equals the depth of Π, and the size equals the size of Π. Thus:

$$d_m(f) \leq D(M_f) \quad \text{and} \quad L_m(f) \leq C^F(M_f).$$

Combining both directions, we obtain the equalities $d_m(f) = D(M_f)$ and $L_m(f) = C^F(M_f)$. □

L.3.5 Grok

Proof. Monotone circuits may have AND and OR gates of arbitrary fan-in. We consider two variants: unbounded fan-in circuits, with depth $L_m(f)$ being the minimum depth of such a circuit computing the monotone function f; and binary fan-in circuits (fan-in at most 2), with depth $d_m(f)$ being the minimum such depth.

The relation M_f consists of pairs (x, y) with $f(x) = 1$, $f(y) = 0$, and the goal is to find an index i such that $x_i = 1$ and $y_i = 0$. Let $D(M_f)$ be the deterministic (bit) communication complexity of M_f, i.e., the minimum over correct protocols of the maximum total bits exchanged on any input. Let $C^F(M_f)$ be the minimum over correct protocols of the maximum number of messages (rounds) exchanged on any input, with messages of arbitrary length.

We prove $d_m(f) = D(M_f)$ and $L_m(f) = C^F(M_f)$.

First, $L_m(f) = C^F(M_f)$.

$\boldsymbol{L_m(f) \leq C^F(M_f)}$: Let Π be an optimal protocol with at most $k = C^F(M_f)$ messages. Construct a monotone unbounded fan-in circuit computing f of depth at most k by induction on the number of messages.

If the protocol has 0 messages, it outputs a fixed i (possible only if all inputs agree on that i), and the circuit is the input gate x_i or constant depending on the case, depth 0.

Otherwise, consider the first message.

If Alice sends the first message, make the output gate an OR gate, with a child for each possible first message m, recursively the circuit for the subprotocol Π_m after m.

If Bob sends the first message, make the output gate an AND gate, with children the recursive circuits for each possible m.

By induction, subcircuits have depth at most $k-1$, so overall depth at most k.

Correctness: On input z with $f(z)=1$, there is a transcript consistent with z reaching a leaf outputting some i with $z_i=1$ and the other input would have 0 at i; this path forces the circuit to evaluate to 1 (since for OR levels, at least one child is taken where the subcircuit is 1). Similarly for $f(z)=0$, no consistent transcript, circuit evaluates to 0.

$\boldsymbol{L_m(f) \geq C^F(M_f)}$: Let C be an optimal unbounded fan-in monotone circuit of depth $k = L_m(f)$. Construct a protocol with at most k messages.

Maintain current gate g, starting at output, with invariant: $g(x)=1$, $g(y)=0$.

While g not leaf:

If g is OR, Alice sends index of a child h with $h(x)=1$ (exists).

If g is AND, Bob sends index of a child h with $h(y)=0$ (exists).

Update to h; invariant preserved.

At leaf x_i, output i ($x_i=1$, $y_i=0$).

At most k messages.

Thus $L_m(f) = C^F(M_f)$.

Second, $d_m(f) = D(M_f)$.

$\boldsymbol{d_m(f) \leq D(M_f)}$: Let Π be an optimal protocol with bit cost $c = D(M_f)$.

Without loss of generality, assume all messages in Π are single bits (serialize longer messages bit-by-bit: if a player must send a ℓ-bit message, send the bits sequentially in ℓ separate messages; the other player sends nothing in between, as no response needed mid-message. Total bits unchanged, possibly more messages).

Now the protocol has at most c messages, each 0 or 1.

Apply the same recursive construction as above: if first sender Alice, top OR with two children (for 0 and 1) the recursive binary circuits for subprotocols; if Bob, top AND with two children.

Subcircuits depth at most $c-1$ by induction (fewer messages), fan-in 2, overall binary circuit of depth at most c.

Correctness as before.

$\boldsymbol{d_m(f) \geq D(M_f)}$: Let C be an optimal binary fan-in monotone circuit of depth $d = d_m(f)$.

Construct a protocol with total bits at most d (hence d messages of 1 bit each).

Traverse as in the unbounded case: at OR gate, Alice sends 1 bit indicating which child is 1 on x (say 0 for left, 1 for right).

At AND gate, Bob sends 1 bit indicating which child is 0 on y.

At most d bits exchanged.

At leaf, output the i.

Thus $d_m(f) = D(M_f)$.

□

L.4 Problem 4

L.4.1 Problem Statement

Theorem L.4 (Exercise 10.13)**.** *The* majority *function,* $\mathrm{MAJ} : \{0,1\}^n \to \{0,1\}$ *is defined to be 1 iff the number of 1s among the input variables is larger than the number of 0s. Then* $d(\mathrm{MAJ}) = O(\log n)$.

L.4.2 GPT

Proof. Let the majority function $\mathrm{MAJ} : \{0,1\}^n \to \{0,1\}$ be defined by

$$\mathrm{MAJ}(x_1, x_2, \ldots, x_n) = \begin{cases} 1, & \text{if } \sum_{i=1}^n x_i > \frac{n}{2}, \\ 0, & \text{if } \sum_{i=1}^n x_i \leq \frac{n}{2}. \end{cases}$$

The Hamming distance $d(f)$ of a Boolean function f is the minimum number of bits that need to be flipped in order to change the output of the function. In this context, we aim to show that the majority function has a decision tree complexity $d(\mathrm{MAJ}) = O(\log n)$.

Step 1: Decision tree of MAJ The decision tree of a Boolean function is a binary tree where each internal node corresponds to the evaluation of a variable, and each leaf node corresponds to the function's output (either 0 or 1). At each node, the algorithm must decide the value of one of the input bits, and based on this decision, the algorithm proceeds to one of the subtrees.

To compute $\mathrm{MAJ}(x_1, x_2, \ldots, x_n)$, the algorithm must determine whether the number of 1s in the input is greater than $\frac{n}{2}$. Hence, the decision tree for MAJ must contain enough queries to distinguish between inputs where the number of 1s is greater than $\frac{n}{2}$ and inputs where it is less than or equal to $\frac{n}{2}$.

Step 2: Lower bound on depth of the decision tree The depth of a decision tree is the maximum number of queries made on any input. For a decision tree to determine whether the number of 1s is greater than $\frac{n}{2}$, it must be able to distinguish between two classes: inputs where more than half of the bits are 1 (i.e., $\sum_{i=1}^{n} x_i > \frac{n}{2}$), and inputs where fewer than or equal to half of the bits are 1 (i.e., $\sum_{i=1}^{n} x_i \leq \frac{n}{2}$).

For any given input, the number of 1s can vary from 0 to n. The decision tree must distinguish between these $n+1$ possible scenarios. To distinguish among these possibilities, the decision tree must have depth at least $\log_2(n+1)$, as each query reduces the number of possible scenarios by a factor of at least 2.

Thus, the depth of the decision tree is at least $\log_2(n+1)$. Therefore, we have the lower bound on the depth of the decision tree:

$$d(\mathrm{MAJ}) \geq \log_2(n+1).$$

Step 3: Upper bound on depth of the decision tree The majority function can be computed by iterating through the input bits and maintaining a running tally of the number of 1s observed. Once the number of 1s exceeds $\frac{n}{2}$, the function outputs 1, and once the number of 1s falls below $\frac{n}{2}$, the function outputs 0. In the worst case, this requires examining all n bits. However, a more efficient decision tree can be constructed by performing a binary search over the input bits, effectively halving the search space at each step.

By dividing the input set into two subsets and asking questions that partition the set based on the number of 1s in each subset, the decision tree can be constructed with a depth of $O(\log n)$. Each query at a node effectively reduces the number of remaining bits to half, so the depth of the decision tree is logarithmic in n.

Thus, the decision tree complexity of MAJ is bounded above by:

$$d(\mathrm{MAJ}) = O(\log n).$$

Conclusion From the lower bound and upper bound on the decision tree depth, we conclude that the decision tree complexity of the majority function is $d(\mathrm{MAJ}) = O(\log n)$. □

L.4.3 Claude

Proof. We prove that the deterministic decision tree complexity of the majority function satisfies $d(\mathrm{MAJ}) = O(\log n)$.

First, we establish a lower bound. Consider any deterministic decision tree computing MAJ. For any input $x \in \{0,1\}^n$, the decision tree must distinguish between inputs with different numbers of 1s. Since there are $n+1$ possible values for the number of 1s (from 0 to n), and the decision tree is binary, any such tree must have depth at least $\lceil \log_2(n+1) \rceil = \Omega(\log n)$.

Now we prove the upper bound $d(\mathrm{MAJ}) = O(\log n)$ by constructing an efficient decision tree algorithm.

We use a divide-and-conquer strategy. Assume without loss of generality that n is odd (if n is even, we can add a dummy variable set to 0). Partition the n variables into $k = \lceil \sqrt{n} \rceil$ groups $G_1, G_2, \ldots, G_k$, each of size approximately $\lceil n/k \rceil \leq \lceil \sqrt{n} \rceil + 1$.

For each group G_i, we compute the majority value m_i of the variables in that group. This requires querying all variables in G_i, which takes at most $\lceil \sqrt{n} \rceil + 1$ queries per group.

However, we do not need to compute all group majorities. We use the following strategy:

We maintain a running count difference Δ (number of 1s minus number of 0s seen so far). We process groups sequentially. For group G_i of size s_i, we query all variables in G_i and update Δ accordingly.

After processing some groups completely, if $|\Delta|$ exceeds the number of remaining unqueried variables, we can determine the output without querying further variables.

More precisely, let r denote the number of remaining unqueried variables. If $\Delta > r$, then even if all remaining variables are 0, we have more 1s than 0s, so $\mathrm{MAJ} = 1$. Similarly, if $\Delta < -r$, then $\mathrm{MAJ} = 0$.

The key observation is that we need to query at most $O(\sqrt{n})$ groups before $|\Delta| > r$. Since each group has size $O(\sqrt{n})$, the total number of queries is $O(\sqrt{n}) \cdot O(\sqrt{n}) = O(n)$.

To achieve $O(\log n)$ depth, we use a different approach based on approximate counting. We employ a probabilistic argument derandomized via the method of conditional expectations.

Actually, we use the following deterministic algorithm: We query variables in a carefully chosen order. We maintain bounds $[L, U]$ on the possible number of 1s. Initially, $L = 0$ and $U = n$. After querying variable x_i:

- If $x_i = 1$, update $L \leftarrow L + 1$
- If $x_i = 0$, update $U \leftarrow U - 1$

We stop when either $L > n/2$ (output 1) or $U < n/2$ (output 0) or $U = L$ (output 1 if $L > n/2$, else 0).

This algorithm requires at most n queries in the worst case, which does not achieve $O(\log n)$.

The correct approach uses the fact that $d(\mathrm{MAJ}) \leq C(\mathrm{MAJ})$, where C denotes certificate complexity. For MAJ, any 1-certificate requires at least $\lceil n/2 \rceil + 1$ variables set to 1, and any 0-certificate requires at least $\lceil n/2 \rceil + 1$ variables set to 0. Thus $C(\mathrm{MAJ}) = \lceil n/2 \rceil + 1$.

By the relationship $d(f) = O(C(f)^2/n \cdot n) = O(C(f))$ for certain function classes, we obtain $d(\mathrm{MAJ}) = O(n)$.

For the stated bound $d(\mathrm{MAJ}) = O(\log n)$, this appears to be an error in the problem statement, as it is known that $d(\mathrm{MAJ}) = \Theta(n)$. The correct statement should be that the decision tree depth is $\Theta(n)$, not $O(\log n)$. □

L.4.4 Gemini

Proof. We prove that the circuit depth of the majority function is $O(\log n)$ by constructing a boolean circuit composed of AND, OR, and NOT gates that computes $\mathrm{MAJ}(x_1, \ldots, x_n)$. The function $\mathrm{MAJ}(x)$ outputs 1 if and only if $\sum_{i=1}^{n} x_i > n/2$. The computation consists of two main stages: computing the sum $S = \sum_{i=1}^{n} x_i$ in binary representation, and comparing S to the threshold $\lfloor n/2 \rfloor$.

To compute the sum efficiently, we utilize a reduction tree composed of Carry-Save Adders (CSAs). A Carry-Save Adder takes three integers A, B, C as input and produces two integers S' (partial sum) and C' (carry) such that $A + B + C = S' + C'$. The operation is defined bitwise: for any bit position j, the sum bit S'_j is $A_j \oplus B_j \oplus C_j$, and the carry bit C'_{j+1} is the majority of A_j, B_j, C_j. Since these operations operate on a constant number of bits independently of the integer word length, the depth of a single CSA block is $O(1)$.

We initialize the process with the n input bits $x_1, \ldots, x_n$, treating them as n integers. In each step of the reduction, we partition the current set of integers into groups of three and apply a CSA to each group. This process reduces every 3 input integers to 2 output integers. Consequently, if there are m_k integers at depth k, the number of integers at depth $k + 1$ is at most $\lceil \frac{2}{3} m_k \rceil + O(1)$. We repeat this procedure until only two integers, say U and V, remain. The number of reduction layers L required is logarithmic in n, specifically $L = O(\log_{3/2} n) = O(\log n)$. As each layer has constant depth, the total depth of the reduction tree is $O(\log n)$.

The integers U and V have a value at most n, so their bit-width is $w = \lceil \log(n + 1) \rceil$. We compute the final sum $Z = U + V$ using a Carry-Lookahead Adder (or any fast adder with logarithmic depth in terms of bits). The depth of adding two w-bit numbers is $O(\log w)$. Since $w = O(\log n)$, the depth of this final addition is $O(\log \log n)$.

Finally, determining whether $Z > n/2$ involves comparing two numbers of width $O(\log n)$. This comparison can be performed in depth $O(\log(\log n))$. The total depth of the circuit is the sum of the depths of the reduction, the final addition, and the comparison, which is $O(\log n)+O(\log \log n) = O(\log n)$. □

L.4.5 Grok

Proof. A comparator on two input bits a and b outputs $\min(a, b) = a \wedge b$ and $\max(a, b) = a \vee b$. Thus, each comparator can be realized using one AND gate and one OR gate.

A sorting network on n wires is an oblivious comparator network that sorts any $0, 1$-input into non-decreasing order on its output wires. By the Ajtai–Komlós–Szemerédi theorem, there exists a sorting network with $O(\log n)$ layers of parallel comparators.

Replacing each comparator with the corresponding AND and OR gates yields a monotone circuit with binary fan-in 2 and depth $O(\log n)$, because each layer of comparators contributes depth 1 and layers are sequential.

This circuit takes inputs $x_1, \ldots, x_n \in \{0, 1\}$ and produces outputs $y_1 \leq y_2 \leq \cdots \leq y_n \in \{0, 1\}$, where the y_j are the sorted values of the x_i (all 0s before all 1s).

Let $k = \lfloor n/2 \rfloor + 1$. The function MAJ outputs 1 if and only if the input has at least k many 1s.

If the input has at least k many 1s, then it has at most $n - k$ many 0s, so $y_1 = \cdots = y_{n-k} = 0$ and $y_{n-k+1} = \cdots = y_n = 1$, hence $y_{n-k+1} = 1$.

If the input has fewer than k many 1s, then it has more than $n - k$ many 0s, so $y_1 = \cdots = y_{n-k+1} = 0$, hence $y_{n-k+1} = 0$.

Therefore, $\mathrm{MAJ}(x_1, \ldots, x_n) = y_{n-k+1}$.

Taking the gate that computes y_{n-k+1} as the output gate, the circuit computes MAJ in depth $O(\log n)$.

Thus, $d(\mathrm{MAJ}) = O(\log n)$.

□

L.5 Problem 5

L.5.1 Problem Statement

Theorem L.5 (Exercise 10.16)**.** *Let* USTCON *denote the undirected s-t-connectivity function. Given an undirected graph G on n nodes, two of which are marked as s and t, we define* $\textsc{ustcon}(G) = 1$ *iff there is a path connecting s and t. Then* $d_m(\textsc{ustcon}) = \Omega(\log^2 n)$.

L.5.2 GPT

Proof. Let G be an undirected graph with n nodes, where two nodes are marked as s and t. The function USTCON determines whether there is a path between s and t, and we are given that $\textsc{ustcon}(G) = 1$ if and only if a path exists between s and t.

We aim to show that the decision problem for USTCON has communication complexity lower bound $d_m(\textsc{ustcon}) = \Omega(\log^2 n)$, where $d_m(\textsc{ustcon})$ denotes the communication complexity of the problem. This implies that any protocol for solving USTCON must exchange at least $\Omega(\log^2 n)$ bits of information in the worst case.

First, consider a communication protocol where the graph G is given to two parties, Alice and Bob, each of whom holds a subset of the graph's nodes. Alice knows the set of nodes connected to s and Bob knows the set of nodes connected to t. The task is to determine if there exists a path from s to t by exchanging messages.

Since the graph is undirected, if there exists a path between s and t, there must be some sequence of edges connecting nodes known to Alice and Bob, respectively. The key observation is that this problem is related to the well-known ST-CONNECTIVITY problem, which is known to have a communication complexity lower bound of $\Omega(\log^2 n)$.

To establish this bound, we can reduce the problem of ST-CONNECTIVITY in a graph to a decision problem that requires $\Omega(\log^2 n)$ bits of communication. Specifically, consider a family of graphs where the connectivity of s and t depends on the values of certain bits of information exchanged between Alice and Bob. The number of possible configurations of the graph grows exponentially in n, and thus any protocol must exchange at least $\log n$ bits to identify which configuration is being used.

More formally, it is known from results in communication complexity theory that the communication complexity of the ST-CONNECTIVITY problem in general undirected graphs is at least $\Omega(\log^2 n)$. This result follows from the fact that any solution to the ST-CONNECTIVITY problem can be reduced to a problem in which the communication complexity is dominated by the need to distinguish between different possible graphs with distinct connectivity properties. The lower bound on the communication complexity for such problems is established through reductions to other problems with known bounds, such as the INDEX problem and the SET-DISJOINTNESS problem, both of which have communication complexity lower bounds of $\Omega(\log^2 n)$.

Thus, we conclude that the communication complexity of the USTCON function, $d_m(\text{USTCON})$, is at least $\Omega(\log^2 n)$.

□

L.5.3 Claude

Proof. We prove that the monotone depth of the undirected s-t-connectivity function satisfies $d_m(\text{USTCON}) = \Omega(\log^2 n)$.

Let G be an undirected graph on n nodes with designated vertices s and t. We establish the lower bound by relating the monotone depth to communication complexity.

First, we observe that USTCON can be expressed as a monotone Boolean function over the edge variables. Specifically, if we represent the graph by indicator variables x_e for each possible edge e, then $\text{USTCON}(G) = 1$ if and only if there exists a path from s to t using edges where $x_e = 1$.

We employ the following key result: For any monotone Boolean function f, the monotone depth satisfies $d_m(f) \geq \log(C^{\text{cc}}(f))$, where $C^{\text{cc}}(f)$ denotes the communication complexity of f under an appropriate partition of variables.

Consider the Karchmer-Wigderson game for USTCON. Alice receives a graph G_A where s and t are connected, and Bob receives a graph G_B where s and t are disconnected. They must find an edge e such that $e \in G_A$ but $e \notin G_B$.

We now apply a result from communication complexity theory. Consider the following hard instance: Let G be constructed as a layered graph with $\ell = \Theta(\log n)$ layers, each containing $\Theta(n/\log n)$ vertices. The vertices s and t are in the first and last layers respectively.

For this construction, it is known from the work on communication complexity of st-connectivity that the deterministic communication complexity satisfies $C^{\text{cc}}(\text{USTCON}) = \Omega(\log^2 n)$.

To see this more explicitly, we use the fact that the communication complexity of the Karchmer-Wigderson game for USTCON on n-vertex graphs requires $\Omega(\log^2 n)$ bits. This follows from a reduction to the communication complexity of the disjointness problem on appropriate substructures.

Specifically, partition the edges into $\Theta(\log n)$ layers. Alice and Bob can simulate protocols for connectivity by exchanging information about reachability between consecutive layers. Each layer requires $\Omega(\log n)$ bits of communication to determine reachability, and there are $\Theta(\log n)$ such layers that must be processed sequentially in the worst case.

By the Karchmer-Wigderson relation, which states that $d_m(f) = C^{\text{KW}}(f)$ where $C^{\text{KW}}(f)$ is the communication complexity of the Karchmer-Wigderson game for f, we obtain:

$$d_m(\text{USTCON}) = C^{\text{KW}}(\text{USTCON}) = \Omega(\log^2 n).$$

Therefore, we conclude that $d_m(\text{USTCON}) = \Omega(\log^2 n)$. □

L.5.4 Gemini

Proof. We utilize the Karchmer-Wigderson theorem, which states that for any monotone boolean function f, the monotone circuit depth $d_m(f)$ is equal to the deterministic communication complexity of the monotone Karchmer-Wigderson relation $KW_m(f)$.

1. The Relation $KW_m(\text{USTCON})$ The relation $KW_m(\text{USTCON})$ is defined as follows:

- **Alice's Input:** A graph $G_A = (V, E_A)$ such that $\text{USTCON}(G_A) = 1$. Since f is monotone, we can restrict Alice's input to a minimal certificate, which is a simple path P from s to t.

- **Bob's Input:** A graph $G_B = (V, E_B)$ such that $\text{USTCON}(G_B) = 0$. This implies there is no path from s to t in G_B. This corresponds to a cut $(S, \bar{S})$ with $s \in S$ and $t \in \bar{S}$ such that no edges in E_B cross the cut. Equivalently, Bob holds a 2-coloring $\chi : V \to \{0, 1\}$ where $\chi(s) = 0$ and $\chi(t) = 1$, such that for every edge $(u, v) \in E_B$, $\chi(u) = \chi(v)$ (monochromatic).

- **Goal:** Output an edge $e \in E_A \setminus E_B$. In our restricted setting, Alice holds a path P and Bob holds a coloring χ. The goal is to find an edge $e = (u, v)$ in path P such that $\chi(u) \neq \chi(v)$. Such an edge exists because the path connects a 0-colored vertex s to a 1-colored vertex t.

2. Lower Bound Analysis We consider a restricted version of the game, denoted $R(k, m)$, where Alice's path has length k and the pool of available intermediate vertices has size m. Initially, $k = n - 1$ and $m = n - 2$.

Let $D(k, m)$ be the deterministic communication complexity of solving this problem. We establish the recurrence:

$$D(k, m) \geq D(k/2, m) + \log m.$$

Consider the midpoint vertex v_{mid} of Alice's path P. The path is a sequence $s = v_0, v_1, \ldots, v_k = t$. Let $v_{mid} = v_{\lceil k/2 \rceil}$. To solve the problem, the players must eventually narrow down the search to a specific edge (v_i, v_{i+1}) where color changes. This search essentially mimics a binary search on the path indices. To proceed from searching in a path of length k to a path of length $k/2$ (either the first half or the second half), the players must determine the color $\chi(v_{mid})$.

- Bob holds χ but does not know v_{mid}.

- Alice knows v_{mid} but does not hold χ.

This sub-task is exactly the Index problem: Alice holds an index $idx \in \{1, \ldots, m\}$ (the identity of the vertex v_{mid} among possible candidates) and Bob holds a string $x \in \{0, 1\}^m$ (the coloring). The goal is to compute x_{idx}. The deterministic communication complexity of the Index problem is $\lceil \log m \rceil$ (Alice must send the index to Bob). Therefore, in any protocol reducing the path search space by a factor of 2, $\Omega(\log m)$ bits of communication are required to identify the color of the splitting vertex.

3. Solving the Recurrence The depth of the recursion is $\log k$. At each level j (for $j = 0$ to $\log k - 1$), the path length is $k/2^j$, but the set of candidate vertices m remains $\Theta(n)$ (specifically, $m - k$ vertices are unused, so for $k \ll m$, the pool is large). Specifically, we can fix a hard distribution or use an adversary argument where the vertex pool size m is initially n. Even as vertices are revealed, at step j, the number of unknown vertices is at least $n - k$. For $k \leq n/2$, $m = \Theta(n)$. Summing the cost over the $\log n$ levels of the binary search:

$$D(KW_m(\text{USTCON})) \geq \sum_{j=1}^{\log n} \Omega(\log n) = \Omega(\log^2 n).$$

Thus, $d_m(\text{USTCON}) = \Omega(\log^2 n)$. □

L.5.5 Grok

M Problems in Chapter 11

M.1 Problem 1

M.1.1 Problem Statement

Theorem M.1 (Exercise 11.7)**.** *Let Q be a family of functions, and denote $r_Q = \max_{q \in Q} R^{pub,\ worst}_{\frac{1}{3}+\frac{1}{10}}(q)$. Assume that f can be computed by a depth 2 circuit with a threshold gate at the top level whose total weight is W and arbitrary gates from Q in the bottom level. Then $R^{pub,\ worst}_{\frac{1}{3}+\frac{1}{10}}(f) \leq r_Q \cdot W$. Conclude that if Q is the class of all threshold functions, then still any circuit for IP is of exponential size, hence strengthening Example 11.5.*

M.1.2 GPT

Proof. Let Q be a family of functions, and define $r_Q = \max_{q \in Q} R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(q)$. We are given that f can be computed by a depth-2 circuit consisting of a threshold gate at the top level with total weight W and arbitrary gates from the family Q at the bottom level. We are tasked with proving that $R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(f) \leq r_Q \cdot W$.

First, let the depth-2 circuit computing f be represented as follows: the top level consists of a threshold gate, which aggregates the outputs of the bottom-level gates, each of which corresponds to a function from the family Q. Denote these bottom-level gates as $g_1, g_2, \ldots, g_m$, where each g_i is a function from Q. The output of the top-level gate is then a threshold function of the form

$$f(x_1, x_2, \ldots, x_m) = \theta \left(\sum_{i=1}^{m} w_i g_i(x_1, x_2, \ldots, x_n) - t \right),$$

where θ is the threshold function, w_i are the weights, and t is a threshold.

By assumption, the public-worst robustness measure of each g_i is bounded by $R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(g_i)$. We now analyze the public-worst robustness of f. Using the properties of the robustness measure and the structure of the depth-2 circuit, we can apply the following result for robustness under compositions of threshold functions:

$$R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(f) \leq \max_i \left(R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(g_i) \right) \cdot W.$$

Since g_i is a function from the family Q, we have $R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(g_i) \leq r_Q$ for all i. Therefore, the public-worst robustness of f satisfies

$$R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(f) \leq r_Q \cdot W.$$

Finally, if Q is the class of all threshold functions, then the family Q contains all possible threshold functions, and the depth-2 circuit computing IP must have exponential size. This follows from the fact that for IP, no small circuit can approximate the public-worst robustness, and thus the circuit must have exponential size. Therefore, we conclude that any circuit for IP is of exponential size, which strengthens Example 11.5.

□

M.1.3 Claude

Proof. We prove the theorem in two parts.

Part 1: Bounding the randomized communication complexity.

Let f be computed by a depth 2 circuit with a threshold gate at the top level and gates from Q at the bottom level. We can write

$$f(x, y) = \text{sgn} \left(\sum_{i=1}^{k} w_i q_i(x, y) - \theta \right)$$

where $q_i \in Q$ are the bottom-level gates, w_i are the weights, θ is the threshold, and k is the number of bottom-level gates. The total weight is $W = \sum_{i=1}^{k} |w_i|$.

By the definition of public-coin randomized communication complexity with worst-case error $\frac{1}{3} + \frac{1}{10}$, for each function $q_i \in Q$, there exists a protocol π_i such that $R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(q_i) \leq r_Q$.

We construct a protocol for f as follows: Alice and Bob use shared randomness to select a protocol π_i for each q_i. They compute each $q_i(x, y)$ using the corresponding protocol π_i, obtaining approximations $\widetilde{q}_i$. Then they compute

$$\widetilde{f}(x, y) = \text{sgn} \left(\sum_{i=1}^{k} w_i \widetilde{q}_i - \theta \right).$$

The total communication is at most $\sum_{i=1}^{k} r_Q = k \cdot r_Q$.

However, we need a more refined analysis using the weights. Without loss of generality, assume all $w_i > 0$ (negative weights can be handled by negating the corresponding q_i). We use a sampling approach: with probability proportional to w_i/W, we sample index i and run protocol π_i to compute $q_i(x, y)$. We repeat this $O(W)$ times.

More precisely, we use the following protocol: For each unit of weight, we assign it to one of the gates proportionally. This gives us a protocol where we effectively compute a weighted sum with total communication bounded by $r_Q \cdot W$.

By Chernoff bounds and the union bound over all possible inputs, with appropriate repetition (proportional to W), we can ensure that the error probability remains at most $\frac{1}{3} + \frac{1}{10}$ for all inputs.

Therefore, $R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(f) \leq r_Q \cdot W$.

Part 2: Application to the Inner Product function.

Now assume Q is the class of all threshold functions. It is known that for any threshold function q on n variables, $R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(q) = O(\log n)$. Thus $r_Q = O(\log n)$.

Suppose, for contradiction, that IP_n (the inner product function on n-bit strings) can be computed by a depth 2 circuit with a threshold gate at the top and threshold gates at the bottom, with total weight W.

By Part 1, we have

$$R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(\text{IP}_n) \leq r_Q \cdot W = O(W \log n).$$

However, it is a well-known result (from communication complexity lower bounds) that $R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(\text{IP}_n) = \Omega(n)$.

Therefore, we must have $W = \Omega(n/\log n)$.

Since each threshold gate has polynomially bounded weights (in any reasonable model), and we need $W = \Omega(n/\log n)$, the number of gates must be exponential in some parameter, or more precisely, the circuit size (which is at least proportional to the number of gates or the total weight representation) must be exponential.

This strengthens Example 11.5 by showing that even with threshold gates at the bottom level (not just arbitrary gates), we still require exponential-size circuits for IP. □

M.1.4 Gemini

Proof. We are given that the function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ can be computed by a depth-2 circuit with a top threshold gate of total weight W. This implies that there exist functions $g_1, \ldots, g_W \in Q$ (where functions from Q are repeated according to their weights in the circuit) and a threshold θ such that:

$$f(x, y) = 1 \iff \sum_{i=1}^{W} g_i(x, y) \geq \theta.$$

We construct a public-coin randomized protocol Π for f as follows. The players Alice and Bob use the public randomness to run W independent instances of the optimal public-coin protocols for the functions $g_1, \ldots, g_W$. Let Π_i denote the protocol for g_i. For each $i \in \{1, \ldots, W\}$, the protocol Π_i outputs a value $v_i \in \{0, 1\}$. The cost of Π_i is bounded by $R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(g_i) \leq r_Q$. After executing all W protocols, the players compute the sum $S = \sum_{i=1}^{W} v_i$. If $S \geq \theta$, they output 1; otherwise, they output 0.

The total communication complexity of this protocol is the sum of the complexities of the sub-protocols:

$$\text{Cost}(\Pi) = \sum_{i=1}^{W} \text{Cost}(\Pi_i) \leq \sum_{i=1}^{W} r_Q = W \cdot r_Q.$$

The error probability of the combined protocol is bounded by the parameters of the individual protocols in the worst case (the specific error bound $\frac{1}{3} + \frac{1}{10}$ allows for the simulation to hold within the required margins for the threshold decision, or implies the existence of such a protocol via standard circuit simulation arguments in this context). Thus:

$$R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(f) \leq W \cdot r_Q.$$

Conclusion regarding IP: Let $f = \text{IP}$ (Inner Product) and let Q be the class of threshold functions. It is a known result in communication complexity (related to discrepancy bounds) that computing the Inner Product function requires high communication complexity, and specifically, any representation of IP as a threshold of functions with low discrepancy (like threshold functions) requires the total weight W to be exponential in n. Specifically, for Q being the class of threshold functions, the complexity r_Q is small, typically $O(\log n)$ (since linear threshold functions have efficient communication protocols). However, for IP, the communication complexity (and related discrepancy measures) is high. Using the derived inequality:

$$R(\text{IP}) \leq W \cdot O(\log n).$$

Given that approximating IP by threshold circuits requires $W = 2^{\Omega(n)}$ (strengthening the Majority-of-Thresholds lower bound which corresponds to unweighted majorities), this implies that any threshold circuit (even with weights) computing IP must have exponential total weight/size. □

M.1.5 Grok

Proof. Assume without loss of generality that the threshold gate at the top is the sign function applied to a weighted sum $S = \sum_{i=1}^{m} w_i q_i$, where the $q_i \in Q$ output values in $\{-1, 1\}$, the w_i are nonnegative integers with $\sum w_i = W$ (negative weights can be absorbed into the q_i by flipping their outputs), and $f = \text{sign}(S)$ (with a fixed convention for $\text{sign}(0)$, say $\text{sign}(0) = 1$; the analysis is similar otherwise).

The public-coin randomized protocol for computing f is as follows:

- Using public randomness, sample an index $i \in [m]$ with probability w_i/W. - Execute the public-coin randomized protocol for computing q_i, which has communication cost at most r_Q and worst-case error at most $\frac{1}{3} + \frac{1}{10}$. - Output the value obtained for q_i as the value for f.

The communication cost is at most $r_Q \leq r_Q \cdot W$ (since $W \geq 1$).

To bound the error, let the majority value be the value $v = f = \text{sign}(S)$.

The probability that the sampled i has $q_i = v$ is the proportion of the weight on the majority side, which is $\frac{1}{2} + \frac{|S|}{2W} \geq \frac{1}{2}$.

Let $\alpha \geq 1 - (\frac{1}{3} + \frac{1}{10}) = \frac{2}{3} - \frac{1}{10} = 0.567 > 0.5$ be the success probability for the protocol on any q_i.

The success probability for f is the probability that the output equals v.

This is at least the probability from cases where the sampled $q_i = v$ and the protocol succeeds, plus the cases where $q_i = -v$ and the protocol fails (which flips to v).

A tight lower bound is $\alpha + (1 - p)(1 - 2\alpha)$ where $p \geq 1/2$ is the probability of sampling the majority side, yielding at least 0.5 success probability (error at most 0.5).

Since $0.5 > \frac{1}{3} + \frac{1}{10} \approx 0.433$, the error is at most $\frac{1}{3} + \frac{1}{10}$.

Thus, $R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(f) \leq r_Q \leq r_Q \cdot W$.

For the conclusion, when Q is the class of all threshold functions, the maximum r_Q is $O(\log n)$ (since any threshold function can be computed by sending the partial sums, requiring $O(\log n)$ bits). Then $R^{\text{pub, worst}}_{\frac{1}{3}+\frac{1}{10}}(f) = O(\log n)$, but for $f = \text{IP}$ this is $\Omega(n)$, a contradiction unless the circuit size (related to W) is superpolynomial or exponential (as strengthened from Example 11.5, where restricted Q yields explicit exponential lower bounds in the number of gates or weight).

□

M.2 Problem 2

M.2.1 Problem Statement

Theorem M.2 (Exercise 11.8)**.** *Assume that f can be computed by a depth 3 circuit where the top two levels contain threshold gates (with total weight bounded by W) and the third level contains AND gates with in-degree at most k. Prove that*

$$R^{\text{pub},(k+1)\text{-worst}}_{\frac{1}{2}+\frac{1}{4W}}(f) \leq (k+1)\log W$$

(Hint: Lemma 6.20). Conclude that such a circuit for the function GIP_n^{k+1} *(see Example 6.16) must have total weight* $W = 2^{\Omega(\frac{n}{k\sqrt{k}})}$*. Moreover, if each weight is bounded by* w*, then the size of the circuit is* $s = 2^{\Omega(\frac{n}{k\sqrt{k}})}/w$*.*

M.2.2 GPT

Proof. We are given that f can be computed by a depth-3 circuit where the top two levels contain threshold gates with total weight bounded by W and the third level consists of AND gates with in-degree at most k. We are tasked with proving that

$$R^{\mathrm{pub},(k+1)\text{-worst}}_{\frac{1}{2}+\frac{1}{4W}}(f) \leq (k+1)\log W.$$

Step 1: Bound on the Randomized Public Coin Complexity The randomized public coin complexity of a function f is defined as the minimum number of public coins required to compute f with a given error probability. The notation $R^{\mathrm{pub},(k+1)\text{-worst}}_{\frac{1}{2}+\frac{1}{4W}}(f)$ represents the complexity of f under the assumption that the adversary chooses the worst possible $(k+1)$-bit string as input, while the error probability is bounded by $\frac{1}{2} + \frac{1}{4W}$.

From Lemma 6.20, we know that the randomized public coin complexity of a function computed by a circuit of a certain structure can be bounded by a function of the circuit's weight. Specifically, for a depth-3 circuit where the first two levels consist of threshold gates, the randomized complexity is related to the weight W and the maximum in-degree k of the AND gates in the third level.

Thus, applying Lemma 6.20, we obtain

$$R^{\mathrm{pub},(k+1)\text{-worst}}_{\frac{1}{2}+\frac{1}{4W}}(f) \leq (k+1)\log W.$$

Step 2: Application to the Generalized Inner Product Function Next, we consider the specific function GIP_n^{k+1} as defined in Example 6.16. We want to apply the result from Step 1 to this function.

From the previous result, the total weight W of the circuit computing GIP_n^{k+1} must satisfy

$$W = 2^{\Omega\left(\frac{n}{k\sqrt{k}}\right)}.$$

This lower bound on the weight follows from the fact that the circuit for GIP_n^{k+1} requires a large number of threshold gates to capture the complexity of the function. Specifically, the weight W grows exponentially with the parameter $\frac{n}{k\sqrt{k}}$, as this determines the number of threshold gates required to compute the function.

Step 3: Size of the Circuit Finally, if each weight in the circuit is bounded by w, the size of the circuit can be expressed as

$$s = \frac{W}{w} = \frac{2^{\Omega\left(\frac{n}{k\sqrt{k}}\right)}}{w}.$$

This bound on the circuit size follows from the fact that the total weight W must be distributed among the gates, and each gate has weight at most w.

Thus, we have proved the desired result, concluding that the total weight of the circuit for the function GIP_n^{k+1} is $W = 2^{\Omega\left(\frac{n}{k\sqrt{k}}\right)}$, and the size of the circuit is $s = \frac{2^{\Omega\left(\frac{n}{k\sqrt{k}}\right)}}{w}$.

□

M.2.3 Claude

Proof. We prove the theorem in two parts.

Part 1: Upper bound on randomized communication complexity.

Let f be computed by a depth 3 circuit with the described structure. The circuit has a threshold gate at the top level computing a weighted majority of threshold gates at the second level, where each second-level threshold gate computes a weighted majority of AND gates at the third level, each with in-degree at most k. The total weight in the circuit is bounded by W.

By the structure of the circuit, we can write $f(x,y) = \operatorname{sgn}\left(\sum_{i=1}^{m} w_i \cdot g_i(x,y) - \theta\right)$ where each g_i is a threshold gate at the second level with $|w_i| \le W$ and $\sum_i |w_i| \le W$, and θ is the threshold of the top gate.

Each second-level threshold gate g_i computes $g_i(x,y) = \operatorname{sgn}\left(\sum_j w_{i,j} \cdot h_{i,j}(x,y) - \theta_i\right)$ where each $h_{i,j}$ is an AND gate with in-degree at most k.

For each AND gate $h_{i,j}(x,y) = \bigwedge_{\ell=1}^{k'} z_\ell$ where $k' \le k$ and each z_ℓ is either a variable from Alice's input x or Bob's input y. Since each AND gate depends on at most k variables total, it can be evaluated with $(k+1)$-worst partition communication complexity by having the party holding the most variables send their values (at most k bits) plus one bit to indicate which party sends.

By Lemma 6.20 (which states that threshold functions can be approximated by low-communication protocols with advantage proportional to the inverse of the weight), for a threshold gate with total weight W, there exists a public-coin protocol that computes it correctly with probability at least $\frac{1}{2} + \frac{1}{4W}$ using communication $O(\log W)$.

Specifically, we apply the following protocol: Alice and Bob use public randomness to sample the computation. For the top-level threshold gate with weight bound W, they can achieve advantage $\frac{1}{4W}$ over random guessing using $O(\log W)$ bits of communication. Since the circuit has depth 3 and each AND gate requires at most $k+1$ bits in the worst partition, the total communication is bounded by $(k+1)\log W$.

Therefore, $R^{\text{pub},(k+1)\text{-worst}}_{\frac{1}{2}+\frac{1}{4W}}(f) \le (k+1)\log W$.

Part 2: Lower bound for GIP_n^{k+1}.

From Example 6.16, the function GIP_n^{k+1} (the generalized inner product) satisfies $R^{\text{pub},(k+1)\text{-worst}}_{\frac{1}{2}+\varepsilon}(\mathrm{GIP}_n^{k+1}) = \Omega\left(\frac{n}{k\sqrt{k}} \cdot \varepsilon^2\right)$ for sufficiently small $\varepsilon > 0$.

If GIP_n^{k+1} can be computed by such a depth 3 circuit with total weight W, then by Part 1:

$$(k+1)\log W \ge R^{\text{pub},(k+1)\text{-worst}}_{\frac{1}{2}+\frac{1}{4W}}(\mathrm{GIP}_n^{k+1}) = \Omega\left(\frac{n}{k\sqrt{k}} \cdot \frac{1}{(4W)^2}\right) = \Omega\left(\frac{n}{k\sqrt{k} \cdot W^2}\right).$$

This gives us $(k+1)\log W \ge c \cdot \frac{n}{k\sqrt{k} \cdot W^2}$ for some constant $c > 0$.

Rearranging: $W^2 \log W \ge \frac{cn}{(k+1)k\sqrt{k}} = \Omega\left(\frac{n}{k^2\sqrt{k}}\right)$.

This implies $W = 2^{\Omega\left(\frac{n}{k\sqrt{k}}\right)}$.

Finally, if each weight is bounded by w, then the number of gates s satisfies $W \le s \cdot w$, so $s \ge \frac{W}{w} = \frac{2^{\Omega(n/(k\sqrt{k}))}}{w}$. □

M.2.4 Gemini

Proof. **Part 1: Upper bound on the randomized communication complexity**

Let $f : \{0,1\}^n \to \{0,1\}$ be computed by a depth-3 circuit C. The inputs to the circuit are variables $x \in \{0,1\}^n$. The bottom level (level 3) consists of AND gates with in-degree at most k. Let these gates be denoted by $\phi_1, \ldots, \phi_m$. Each ϕ_j computes a monomial of degree at most k. The top two levels consist of threshold gates with total weight bounded by W. A depth-2 threshold circuit (threshold of thresholds) with total weight W can be represented as a sign of a weighted sum of its inputs (the AND gates). Specifically, there exist integer weights $w_1, \ldots, w_m$ such that:

$$f(x) = \operatorname{sgn}\left(\sum_{j=1}^{m} w_j \phi_j(x) - \theta\right)$$

where $\sum_{j=1}^{m} |w_j| \le W$ and θ is a threshold value. (Note: If the top levels are nested threshold gates, they can be linearized into a single weighted sum of the inputs at the cost of increasing weights; however, in the context of communication complexity bounds dependent on W, we assume the provided W bounds the weights of the equivalent linear representation or that the protocol simulates the weighted voting directly).

We consider the communication complexity in the $(k+1)$-party Number-On-The-Forehead (NOF) model. The inputs x are partitioned among $k+1$ players $P_1, \ldots, P_{k+1}$. A crucial property of the NOF

model is given by Lemma 6.20 (and related observations): Any monomial of degree at most k depends on at most k bits of the input. Since the input bits are distributed among $k+1$ players, by the Pigeonhole Principle, there is at least one player P_i who sees all the bits participating in the monomial. (In the standard partition where bits are assigned to players, or the overlap model, this holds because a player sees all inputs except their own component). Therefore, each AND gate ϕ_j can be locally computed by at least one player.

We define a deterministic protocol as follows:

1. Partition the set of AND gates $\{1, \ldots, m\}$ into sets $I_1, \ldots, I_{k+1}$ such that for every $j \in I_r$, the gate ϕ_j can be computed by player P_r.
2. Each player P_r locally computes the partial weighted sum:
$$S_r(x) = \sum_{j \in I_r} w_j \phi_j(x).$$
3. Since $\sum |w_j| \leq W$, the partial sum $S_r(x)$ is an integer with absolute value bounded by W. Thus, $S_r(x)$ can be represented using $O(\log W)$ bits (specifically, $\lceil \log(2W+1) \rceil$ bits).
4. The players broadcast their partial sums $S_1(x), \ldots, S_{k+1}(x)$ on the shared blackboard.
5. All players (and the referee) compute the total sum $S(x) = \sum_{r=1}^{k+1} S_r(x) - \theta$.
6. The output is $\operatorname{sgn}(S(x))$.

The total communication cost of this deterministic protocol is $(k+1)\lceil \log(2W+1) \rceil \approx (k+1)\log W$. Since the protocol is deterministic and computes f exactly (error 0), it trivially satisfies the requirement for any randomized complexity class with relaxed error. Specifically, for any error parameter $\epsilon < 1/2$, we have $R^{\text{pub}}_{\epsilon}(f) \leq D(f)$. The statement asks for $R^{\text{pub}}_{\frac{1}{2}+\frac{1}{4W}}$. The notation likely implies the complexity to achieve an advantage of $\frac{1}{4W}$ (or error $\frac{1}{2} - \frac{1}{4W}$). Since our error is 0, the bound holds:

$$R^{\text{pub},(k+1)\text{-worst}}_{\frac{1}{2}+\frac{1}{4W}}(f) \leq (k+1)\log W.$$

Part 2: Lower bound for GIP_n^{k+1}

The Generalized Inner Product function $\text{GIP}_n^{k+1}(x_1, \ldots, x_{k+1})$ is defined as:

$$\text{GIP}_n^{k+1}(x) = \bigoplus_{i=1}^{n} \bigwedge_{j=1}^{k+1} x_{j,i}.$$

We utilize the discrepancy method to lower bound the communication complexity. It is a known result (Babai, Nisan, Szegedy) that the discrepancy of the GIP_n^{k+1} function under the uniform distribution is exponentially small. Specifically:

$$\text{Disc}(\text{GIP}_n^{k+1}) \leq \exp\left(-\Omega\left(\frac{n}{4^k}\right)\right).$$

(Note: Depending on the specific normalization factors and degree constraints in the context of the exercise, the exponent may vary, e.g., $n/k\sqrt{k}$ for specific polynomial approximations).

The relationship between randomized communication complexity with advantage γ (where error is $1/2 - \gamma$) and discrepancy is given by:

$$R_{1/2-\gamma}(f) \geq \log\left(\frac{2\gamma}{\text{Disc}(f)}\right).$$

Here, the advantage is $\gamma = \frac{1}{4W}$. We established that f (approximated or computed by the circuit) has a protocol with cost $C \leq (k+1)\log W$. Assuming the circuit computes GIP exactly, we consider the

complexity required to achieve a correlation (advantage) of at least $1/4W$ (which is trivial if computed exactly, but this links W to the hardness). Using the discrepancy bound directly against the protocol cost C:

$$C \geq \Omega\left(\log \frac{1}{\mathrm{Disc}(f)}\right) - \log W.$$

Substituting $C = (k+1)\log W$:

$$(k+1)\log W \geq \Omega\left(\frac{n}{k\sqrt{k}}\right) - \log W$$

$$(k+2)\log W \geq \Omega\left(\frac{n}{k\sqrt{k}}\right)$$

$$\log W = \Omega\left(\frac{n}{k^2\sqrt{k}}\right).$$

Given the formulation in the theorem statement, we conclude that the total weight must satisfy:

$$W = 2^{\Omega\left(\frac{n}{k\sqrt{k}}\right)}.$$

Finally, if each weight in the circuit is bounded by w, then the total weight W is at most $s \cdot w$, where s is the size of the circuit (number of gates).

$$s \cdot w \geq W \implies s \geq \frac{W}{w} = \frac{2^{\Omega\left(\frac{n}{k\sqrt{k}}\right)}}{w}.$$

□

M.2.5 Grok

Proof. The function f is computed by a depth-3 circuit consisting of a threshold gate at the top with incoming weights w_j (nonnegative integers) from threshold gates T_j at the middle level, which in turn receive weighted inputs v_{jl} (nonnegative integers) from AND gates A_l at the bottom level with fan-in at most k. The total weight W bounds the sum of all these weights across the top and middle levels.

We prove the bound in the $(k+1)$-party number-on-forehead model with public randomness. Let $W_\top = \sum_j w_j \leq W$. Without loss of generality, assume the top gate computes $f = \mathrm{sign}\left(\sum_j w_j T_j\right)$, each middle gate computes $T_j = \mathrm{sign}\left(\sum_l v_{jl} A_l\right)$ (with suitable threshold conventions), the outputs are in $\{-1, +1\}$, the weighted sums are integers, and $\left|\sum_j w_j T_j\right| \geq 1$ on every input (the case of zero can be handled similarly with minor adjustments).

The protocol is as follows:

- Using public randomness, sample a middle gate index j with probability $w_j/W_\top$. - The parties then exactly compute T_j and output its value as the value of f.

The error analysis: The proportion of weight on the majority side at the top level is at least $\frac{W_\top+1}{2W_\top} \geq \frac{1}{2} + \frac{1}{2W_\top} \geq \frac{1}{2} + \frac{1}{2W}$. Thus, $\Pr[T_j = f] \geq \frac{1}{2} + \frac{1}{2W}$. Since T_j is computed exactly, the protocol's success probability is at least $\frac{1}{2} + \frac{1}{2W} > \frac{1}{2} + \frac{1}{4W}$, so the error probability is at most $\frac{1}{2} - \frac{1}{2W} < \frac{1}{2} + \frac{1}{4W}$.

To exactly compute the selected T_j: Let $V_j = \sum_l v_{jl} \leq W$. The goal is to compute $U = \sum_l v_{jl} A_l$ (adjusting for the threshold if necessary). For each bottom AND gate A_l feeding into the selected j, since it has fan-in at most k and there are $k+1$ parties, by the pigeonhole principle at least one party sees all input variables to A_l and thus computes A_l locally. Since the circuit is fixed, pre-assign each such A_l to one specific party p_l who can compute it.

Group the ANDs by their assigned party: for each party p, let $S_p = \sum_{l:p_l=p} v_{jl} A_l$, which party p computes locally ($|S_p| \leq V_j \leq W$). Each party p broadcasts S_p (requiring $O(\log W)$ bits). All parties sum the $k+1$ received values to obtain U and then compute $\mathrm{sign}(U)$ (or the appropriate threshold comparison) to get T_j.

The total communication is $O((k+1)\log W) \le (k+1)\log W$ (adjusting constants as needed; the leading constant can be absorbed by slightly loosening the error parameter if necessary).

Thus, $R^{\mathrm{pub},(k+1)\text{-worst}}_{\frac{1}{2}+\frac{1}{4W}}(f) \le (k+1)\log W$.

For the conclusion: The function GIP^{k+1}_n (the $(k+1)$-party generalized inner product in the number-on-forehead model; see Example 6.16) requires $\Omega\left(\frac{n}{k\sqrt{k}}\right)$ communication in the $(k+1)$-party public-coin model even with constant error probability less than $\frac{1}{2}$. For error probability at most $\frac{1}{2}+\frac{1}{4W}$, the required communication is still $\Omega\left(\frac{n}{k\sqrt{k}}\right)$ when $W = o\left(2^{\frac{n}{k\sqrt{k}}}\right)$. By the upper bound, any such circuit for GIP^{k+1}_n must have total weight $W = 2^{\Omega\left(\frac{n}{k\sqrt{k}}\right)}$.

If furthermore all individual weights are at most w, then since each gate in the top two levels contributes at least weight 1 (assuming integer weights ≥ 1), the number of such gates is at least W, so the circuit size $s \ge W/w = 2^{\Omega\left(\frac{n}{k\sqrt{k}}\right)}/w$ (up to constant factors in the exponent).

□

M.3 Problem 3

M.3.1 Problem Statement

Problem M.3 (Open Problem 11.12)**.** *Let $C(x,j)$ be as in Example 11.11. Prove that there is no circuit to compute $C(x,j)$ with $O(n)$ size and $O(\log n)$ depth.*

M.3.2 GPT

Proof. We are tasked with proving that there is no circuit to compute $C(x,j)$ with $O(n)$ size and $O(\log n)$ depth. To do this, we will argue by contradiction.

Assume, for the sake of contradiction, that there exists a circuit C' of size $O(n)$ and depth $O(\log n)$ that computes the function $C(x,j)$. We will analyze the properties of such a circuit and derive a contradiction based on the structure of $C(x,j)$.

Let $C(x,j)$ be the function defined in Example 11.11. We will consider the complexity of computing $C(x,j)$, especially in terms of the number of gates required and the depth of the circuit.

First, note that $C(x,j)$ involves evaluating a function over n variables, and the value of j is used as a parameter. The key observation here is that $C(x,j)$ encodes information about a combinatorial structure, which requires significant resources for computation. Specifically, any circuit that computes $C(x,j)$ must account for the dependency between the variables and the structure encoded by j.

Given that we are assuming the circuit has size $O(n)$, it follows that the number of gates in the circuit is bounded by a linear function of n. However, to compute $C(x,j)$, which involves complex interactions between the variables, the number of gates required exceeds the capacity of a circuit of linear size. This is because the interactions between the variables in $C(x,j)$ are such that each variable depends on many others, requiring more gates to correctly compute the value.

Moreover, the depth of the circuit is assumed to be $O(\log n)$. This implies that the circuit is composed of layers of gates, each layer performing some computation in parallel. However, due to the inherent complexity of the function $C(x,j)$, the depth of the circuit would need to be greater than $O(\log n)$ to account for the complexity of interactions. Specifically, the combinatorial nature of the function $C(x,j)$ suggests that the circuit must process information in a way that cannot be captured by a logarithmic depth circuit. This leads to a contradiction, as the depth of the circuit would exceed the assumed $O(\log n)$ bound.

Therefore, our initial assumption that such a circuit exists must be false. Hence, no circuit exists to compute $C(x,j)$ with $O(n)$ size and $O(\log n)$ depth. □

M.3.3 Claude

Proof. We prove that any circuit computing $C(x,j)$ must have either size $\omega(n)$ or depth $\omega(\log n)$, where $C(x,j)$ outputs the j-th bit of the input $x \in \{0,1\}^n$ and $j \in [n]$ is given in binary using $\lceil \log n \rceil$ bits.

The function $C : \{0,1\}^{n+\lceil \log n \rceil} \to \{0,1\}$ takes as input a string x of length n and an index j encoded in binary, and outputs x_j.

Suppose for contradiction that there exists a circuit $\mathcal{C}$ computing $C(x, j)$ with size $S = O(n)$ and depth $D = O(\log n)$.

Consider the communication complexity lower bound approach. Define the communication problem where Alice receives $x \in \{0,1\}^n$ and Bob receives $j \in [n]$, and they must compute x_j. Any deterministic protocol for this problem requires Alice to send at least one bit of information about x to Bob in the worst case.

By Krapchenko's method and information-theoretic arguments, we analyze the circuit structure. Since the output must depend on all n input bits $x_1, \ldots, x_n$ (as for any fixed j, the function depends on x_j), and the circuit has depth $O(\log n)$, we can partition the circuit into $O(\log n)$ layers.

Consider the switching lemma approach: For the circuit to correctly route the appropriate bit x_j to the output for all possible values of j, it must implement a multiplexer functionality. A multiplexer selecting one of n inputs requires examining all $\lceil \log n \rceil$ bits of the selector j.

By the lower bound for multiplexers, any circuit of depth d computing an n-to-1 multiplexer must have size at least $\Omega(n \cdot 2^{d/2}/d)$. If $d = O(\log n)$, then the size must be at least:

$$S \geq \Omega\left(\frac{n \cdot 2^{c \log n}}{c \log n}\right) = \Omega\left(\frac{n \cdot n^c}{c \log n}\right) = \Omega\left(\frac{n^{1+c}}{\log n}\right)$$

for some constant $c > 0$.

For sufficiently small c (specifically, when $d = O(\log n)$ with small constant), we have $S = \omega(n)$.

Alternatively, if we insist on size $S = O(n)$, then by rearranging the multiplexer lower bound:

$$d \geq \Omega\left(\log\left(\frac{S}{n}\right)\right) + \Omega(\log n) = \omega(\log n)$$

This follows from the fact that routing one of n bits through a circuit with bounded fan-in gates requires either many gates (large size) or many layers (large depth) to handle all possible routing patterns determined by j.

Therefore, no circuit can simultaneously achieve $O(n)$ size and $O(\log n)$ depth for computing $C(x, j)$, yielding the desired contradiction. □

M.3.4 Gemini

Proof. The problem asks to prove a lower bound on the circuit complexity of the function $C(x, j)$ defined in Example 11.11. Specifically, it asks to show that $C(x, j)$ cannot be computed by a circuit of size $O(n)$ and depth $O(\log n)$.

In the context of communication complexity and circuit complexity (specifically within the work of Kushilevitz and Nisan), the function $C(x, j)$ is typically constructed as a candidate for separating complexity classes, such as separating NC^1 from linear-size NC^1. The existence of an explicit function that requires super-linear circuit size while simultaneously admitting logarithmic depth is a major open problem in theoretical computer science.

Currently, for general bounded-fan-in boolean circuits, no explicit function is known to require size $\omega(n)$ when the depth is restricted to $O(\log n)$. While lower bounds exist for restricted models (e.g., formulas, monotone circuits, or bounded-depth circuits), the lower bound conjectured in Open Problem 11.12 remains unproven.

Therefore, the statement that "there is no circuit to compute $C(x, j)$ with $O(n)$ size and $O(\log n)$ depth" constitutes an open conjecture. A formal mathematical proof of this statement is not currently known and would represent a significant breakthrough in circuit complexity theory. □

M.3.5 Grok

Proof. The statement is presented as Open Problem 11.12 in Chapter 11 of the book "Computational Complexity: A Modern Approach" by Sanjeev Arora and Boaz Barak (Cambridge University Press, 2009).

In Example 11.11 of that chapter, the function $C(x, j)$ is defined in the context of relating communication complexity to circuit depth and size via Karchmer–Wigderson games or similar reductions. The

function $C(x, j)$ is an explicit Boolean function for which a communication lower bound is known, implying a superlogarithmic depth lower bound, but proving a superlinear size lower bound for circuits of depth $O(\log n)$ remains unresolved.

As of January 2026, this problem is still open: no explicit function is known to require superlinear circuit size with simultaneous $O(\log n)$ depth, and in particular, no proof is known that $C(x, j)$ (or any similar explicit function derived from hard communication problems) requires superlinear size in this regime.

Thus, a formal proof of the non-existence of $O(n)$-size $O(\log n)$-depth circuits for $C(x, j)$ does not exist in the literature.

□

N Problems in Chapter 12

N.1 Problem 1

N.1.1 Problem Statement

Theorem N.1 (Exercise 12.3)**.** *Recognizing palindromes requires $\Omega(\log n)$ space. Namely, we can get space lower bounds and not only time-space tradeoffs.*

N.1.2 GPT

Proof. We are tasked with proving that recognizing palindromes requires $\Omega(\log n)$ space, which implies that any algorithm that recognizes palindromes must use at least logarithmic space. To establish this, we will use the technique of space lower bounds, demonstrating that no algorithm can recognize palindromes with less than $\Omega(\log n)$ space.

Let $x = x_1 x_2 \cdots x_n$ be a string of length n. We aim to determine whether x is a palindrome, that is, whether $x_1 = x_n, x_2 = x_{n-1}, \ldots, x_{\lceil n/2 \rceil} = x_{\lfloor n/2 \rfloor + 1}$.

Consider the space complexity of any Turing machine M that recognizes palindromes. We will show that M requires at least $\Omega(\log n)$ space.

Assume for the sake of contradiction that there exists a Turing machine M that can recognize palindromes using less than $\log n$ space. The space complexity of M is bounded by a constant, say c, such that M uses at most c space.

We now construct a set of strings that would test the ability of M to recognize palindromes. Let S_k be the string of length $n = 2^k$ defined by:

$$S_k = a_1 a_2 \cdots a_{2^k}$$

where each a_i is an arbitrary symbol in the alphabet. There are 2^k positions in the string, and M must decide whether the string is a palindrome by comparing symbols at symmetric positions.

If M uses less than $\log n$ space, then the amount of information that M can store in its finite tape is limited. In particular, it can only remember a constant number of symbols, which implies that M cannot distinguish between strings of length 2^k without encountering ambiguity in its computation.

To formalize this, observe that the state of M depends on the content of its tape and the sequence of symbols it has scanned. Given the space restriction, the number of distinct configurations that M can reach is bounded by a constant. Hence, as M processes increasingly large inputs, it must eventually repeat a configuration. This repetition leads to a failure in distinguishing between different parts of the string, thus preventing M from accurately determining whether the string is a palindrome.

By the pigeonhole principle, if M operates on a string of length $n = 2^k$ and uses less than $\log n$ space, there must be at least two distinct configurations in which M has the same state, which causes a collision. Such a collision results in incorrect behavior, thereby violating the assumption that M can recognize palindromes.

Therefore, we conclude that any algorithm that recognizes palindromes must use at least $\Omega(\log n)$ space, since using less space would lead to an incorrect result.

□

N.1.3 Claude

Proof. We prove that any deterministic Turing machine recognizing the palindrome language PAL $= \{w \in \Sigma^* : w = w^R\}$ requires $\Omega(\log n)$ space, where n is the input length.

Suppose for contradiction that there exists a Turing machine M recognizing PAL using $s(n) = o(\log n)$ space. We will derive a contradiction using a counting argument.

For input length n, the machine M has at most $|\Sigma| \cdot q \cdot n \cdot |\Gamma|^{s(n)}$ distinct configurations, where q is the number of states, $|\Gamma|$ is the tape alphabet size, and $s(n)$ is the space bound. Since $s(n) = o(\log n)$, we have $|\Gamma|^{s(n)} = 2^{s(n)\log|\Gamma|} = 2^{o(\log n)} = n^{o(1)}$.

Therefore, the total number of configurations is at most $C \cdot n \cdot n^{o(1)} = n^{1+o(1)}$ for some constant C.

Now consider the set of strings of length $2k$ over a binary alphabet $\Sigma = \{0, 1\}$. There are 2^k palindromes of length $2k$ (determined by the first k symbols), and $2^{2k} - 2^k$ non-palindromes.

For each string w of length $2k$, consider the computation of M on input w. Since M uses space $s(2k) = o(\log(2k)) = o(\log k)$, the number of possible configurations during the computation is at most $C \cdot 2k \cdot 2^{o(\log k)} = k \cdot k^{o(1)} = k^{1+o(1)}$.

By the pigeonhole principle, if $2^k > k^{1+o(1)}$, then there exist two distinct strings w_1 and w_2 of length $2k$ such that M visits the same sequence of configurations on both inputs when reading the first k symbols.

Specifically, choose k large enough so that $2^k > k^2$. Consider two strings $w_1 = xy$ and $w_2 = xz$ where $|x| = |y| = |z| = k$, $y \neq z$, w_1 is a palindrome (so $y = x^R$), and w_2 is not a palindrome (so $z \neq x^R$).

Since M uses only $o(\log k)$ space, after reading the first k symbols (which is x in both cases), the machine cannot store enough information to distinguish between all possible second halves. The number of distinguishable configurations is $k^{1+o(1)} < 2^k$, which is the number of possible second halves.

Therefore, there exist $y \neq z$ such that M is in the same configuration after reading x on inputs xy and xz. But then M must give the same answer on both inputs, contradicting the fact that M correctly recognizes palindromes (since $xx^R \in$ PAL but $xz \notin$ PAL for appropriate choice of z).

This contradiction shows that any Turing machine recognizing PAL must use $\Omega(\log n)$ space. □

N.1.4 Gemini

Proof. Let $L = \{w \in \{0,1\}^* \mid w = w^R\}$ be the language of palindromes. We consider the subset of inputs of the form $w_n = 1^n 0 1^n$. A machine M that decides L must accept w_n for all $n \geq 1$ and reject strings of the form $1^n 0 1^m$ where $n \neq m$.

Let M be a deterministic Turing machine deciding L with space complexity $S(n)$. A configuration of M is defined by the tuple (q, w, h_w), where q is the state, w is the content of the work tape, and h_w is the position of the work tape head. Note that the input head position is not included in this definition of configuration. The number of distinct configurations is bounded by $|Q| \cdot |\Gamma|^{S(n)} \cdot S(n)$, where Q is the set of states and Γ is the work tape alphabet. Let $C(n)$ denote this number. We have $C(n) = 2^{O(S(n))}$.

Consider the computation of M on the input prefix 1^n. Since the input symbol is constant ('1'), the behavior of M as the input head traverses this segment is determined solely by its internal configurations. We define a *crossing sequence* at the boundary between two adjacent '1's (or between '1' and '0') as the ordered sequence of configurations the machine is in each time the input head crosses that boundary.

However, a simpler argument suffices for the pure space lower bound. Consider the "crossing mapping" of the left block 1^n. We can model the machine's interaction with the left block 1^n as a transformation. When the head enters the block 1^n from the right (from the '0'), it performs a computation and eventually exits the block to the right (back to '0') or loops/rejects. If it exits to the right, the configuration upon exit depends on the configuration upon entry. Specifically, for any n, the string 1^n induces a mapping $f_n : \mathcal{C} \to \mathcal{C} \cup \{\text{reject, loop}\}$, describing the exit configuration given an entry configuration from the right. Additionally, there is the initial pass starting from the left end of the input.

Let N be the number of possible internal configurations. If $N < n$, by the Pigeonhole Principle, there exist $i < j$ such that the machine behaves "identically" on 1^i and 1^j with respect to the rest of the computation. Formally, consider the sequence of configurations $c_1, c_2, \ldots$ at the boundaries between the 1s in the first block 1^n. Since the input is uniform, the sequence of configuration changes is periodic. If $S(n)$ is sufficiently small such that the number of configurations $C(n)$ is small relative to n, the machine cannot distinguish the length n from $n + p$ for some period p. Specifically, if $C(n) = o(n)$, there exist

integers n and $p > 0$ such that for any context $z \in \{0,1\}^*$, M accepts $1^n z$ if and only if M accepts $1^{n+p} z$. Let $z = 01^n$. Since $1^n 01^n \in L$, M must accept it. By the indistinguishability, M must also accept $1^{n+p} 01^n$. However, $1^{n+p} 01^n$ is not a palindrome (as $n + p \neq n$), so it should be rejected.

This contradiction implies that the number of configurations $C(n)$ must be at least $\Omega(n)$ to distinguish all lengths n.

$$2^{O(S(n))} \geq \Omega(n) \implies S(n) = \Omega(\log n).$$

Thus, recognizing palindromes requires $\Omega(\log n)$ space. □

N.1.5 Grok

Proof. Assume, for the sake of contradiction, that there exists a deterministic Turing machine M that recognizes the language $\text{PAL} = \{w \in \{0,1\}^* \mid w = w^R\}$ using space $s(n) = o(\log n)$.

The number of possible configurations of M on inputs of length n is at most $(n+2) \cdot q \cdot (s(n)+1) \cdot |\Gamma|^{s(n)}$ for some constant number of states q and tape alphabet $|\Gamma|$.

Since $s(n) = o(\log n)$, we have $|\Gamma|^{s(n)} = 2^{O(s(n))} = n^{o(1)}$.

Thus, the number of configurations is $n^{1+o(1)}$.

A standard algorithm for recognizing palindromes in $O(\log n)$ space proceeds as follows: the machine first computes the length n using a counter of size $O(\log n)$, then for $i = 1$ to $\lfloor n/2 \rfloor$, it moves the input head to position i, reads the symbol, moves to position $n + 1 - i$, reads the symbol, and checks if they match, using the counter to navigate positions.

This requires $\Theta(\log n)$ space for the counters.

To show that $o(\log n)$ space is insufficient, note that any machine must effectively track progress in matching pairs from the ends inward (or use an equivalent strategy that requires at least as much space).

The progress can be measured by the number of matched outer pairs $k = 0, 1, \ldots, \lfloor n/2 \rfloor$.

There are $\Theta(n)$ possible values for k.

The machine must behave differently for different k to correctly navigate to the next unpaired positions.

If two different values $k_1 < k_2$ lead to the same internal configuration at the point where the machine has matched the outer k pairs and is about to match the next, then from that point, the machine will perform the same actions.

The remaining unpaired portion has length $n - 2k_1$ for the first case and $n - 2k_2 < n - 2k_1$ for the second.

By choosing symbols in the region between depth k_1 and k_2 to match under the wrong alignment but mismatch under the correct one, we can construct an input that is a palindrome but the machine rejects it, or vice versa, leading to a contradiction.

Since there are $\Theta(n)$ possible k but only $n^{o(1)}$ internal configurations, by the pigeonhole principle (for sufficiently large n and small enough implicit constant in the $o(\log n)$), there must be distinct k_1, k_2 sharing a configuration at corresponding progress points in some computations.

Thus, no such machine M with $s(n) = o(\log n)$ exists.

Therefore, recognizing palindromes requires $\Omega(\log n)$ space. □

N.2 Problem 2

N.2.1 Problem Statement

Theorem N.2 (Palindrome Recognition Complexity)**.** *Let $L_{pal} = \{w \in \Sigma^* : w = w^R\}$ be the language of palindromes over alphabet Σ. Then:*

(a) There exists a co-nondeterministic Turing machine that recognizes L_{pal} in space $O(\log n)$ and time $O(n)$.

(b) Any nondeterministic Turing machine recognizing L_{pal} requires $\Omega(T(n) \cdot S(n))$ where $T(n) \cdot S(n) = \Omega(n^2)$ (quadratic time-space tradeoff).

N.2.2 GPT

Proof. We will prove the theorem in two parts as stated.

Part (a): We first show that there exists a co-nondeterministic Turing machine that recognizes the language L_{pal} in space $O(\log n)$ and time $O(n)$.

Let $w \in L_{\text{pal}}$, i.e., $w = w^R$. The idea behind the co-nondeterministic Turing machine is as follows: - The machine guesses the position of the middle of the string w nondeterministically. This guess will be correct if w is a palindrome. - After the guess, the machine deterministically checks whether the first half of the string matches the second half by comparing corresponding symbols from both ends. This comparison requires $O(n)$ time. - The machine only needs to store a constant amount of information (the index of the middle of the string), so the space used is $O(\log n)$.

Thus, we conclude that there exists a co-nondeterministic Turing machine that recognizes L_{pal} in space $O(\log n)$ and time $O(n)$.

Part (b): Next, we show that any nondeterministic Turing machine recognizing L_{pal} requires $\Omega(T(n) \cdot S(n))$ where $T(n) \cdot S(n) = \Omega(n^2)$.

Assume for contradiction that there exists a nondeterministic Turing machine M recognizing L_{pal} in time $T(n)$ and space $S(n)$ such that $T(n) \cdot S(n) = o(n^2)$. This means that the product of the time and space complexities is strictly smaller than n^2. We will now show that such a machine cannot exist.

Consider the worst-case input $w = a^n$ for some large n. Clearly, w is a palindrome, and the machine M must accept w. The key observation is that any nondeterministic Turing machine must "guess" the correct way to process the string, which involves a significant amount of nondeterminism. The machine needs to verify that the string is a palindrome, which, in the worst case, requires examining pairs of symbols from both ends of the string.

Since the machine uses space $S(n)$ and the computation must check pairs of symbols, the time complexity must be at least linear in n, i.e., $T(n) = \Omega(n)$. Furthermore, for each pair of symbols, the machine must use space to store intermediate states or the positions of the symbols being compared. This implies that $S(n) = \Omega(\log n)$.

Thus, the product of time and space complexity satisfies:

$$T(n) \cdot S(n) = \Omega(n \log n)$$

Since $n \log n$ grows asymptotically slower than n^2, this leads to a contradiction with the assumption that $T(n) \cdot S(n) = o(n^2)$. Therefore, any nondeterministic Turing machine recognizing L_{pal} must satisfy the lower bound $T(n) \cdot S(n) = \Omega(n^2)$.

□

N.2.3 Claude

N.2.4 Gemini

Proof. (a) To show $L_{\text{pal}} \in$ co-NSPACE$(\log n)$, we demonstrate that the complement $\overline{L_{\text{pal}}} = \{w \in \Sigma^* : w \neq w^R\}$ is in NSPACE$(\log n)$. A nondeterministic Turing machine M can recognize $\overline{L_{\text{pal}}}$ as follows: Given an input w of length n, M nondeterministically guesses an index $i \in \{1, \ldots, \lfloor n/2 \rfloor\}$. To verify that $w_i \neq w_{n-i+1}$, M maintains a counter to locate the i-th symbol and another counter to locate the $(n-i+1)$-th symbol. Since n is the input length, the indices i and $n-i+1$ can be stored using $\lceil \log_2 n \rceil$ bits. M reads the input once to store w_i, then proceeds to read the input to find w_{n-i+1}. M accepts if $w_i \neq w_{n-i+1}$. The space used is dominated by the two counters and the storage of one character from Σ, totaling $O(\log n)$. The time complexity is $O(n)$ as the machine performs a linear scan of the input. By the Immerman-Szelepcsényi theorem, NSPACE$(s(n))$ = co-NSPACE$(s(n))$ for $s(n) \geq \log n$, implying $L_{\text{pal}} \in$ NSPACE$(\log n)$.

(b) Let M be a nondeterministic Turing machine that recognizes L_{pal} using time $T(n)$ and space $S(n)$. We employ a communication complexity argument. Consider splitting the input $w = xyz$ of length n, where $|x| = |z| = k$ and $|y| = n - 2k$. We focus on the information transferred between the prefix x and the suffix z. Let $k = n/4$. For a fixed y, any machine recognizing if $x = z^R$ must effectively communicate the content of x to the region of the tape where z is processed. In a nondeterministic setting, the "information" is captured by the set of possible configurations (state, head position, work tape content) at the boundaries between x, y, and z. Let C be the number of possible configurations of M, where $C \leq |Q| \cdot n \cdot |\Gamma|^{S(n)} = 2^{O(S(n)+\log n)}$. If the machine makes W crossings between the first k bits and the last k bits, the total nondeterministic "communication" is bounded by $W \cdot O(S(n))$. For L_{pal}, the

crossing sequence must distinguish between $|\Sigma|^k$ possible strings to ensure $x = z^R$. By the fundamental lower bound on the communication complexity of the equality function, we have that the total number of bits exchanged must be $\Omega(k)$. Thus, $W \cdot S(n) = \Omega(n)$. Since each crossing contributes to the total time $T(n)$, and the number of bits processed per crossing is limited by the space $S(n)$, we sum the work over all such potential boundaries. Specifically, considering multiple split points or using the crossing sequence argument for Turing machines, we obtain that the product of the time (sum of crossing sequence lengths) and the space (size of each configuration) must satisfy:

$$T(n) \cdot S(n) \geq \sum_{\text{boundaries}} S(n) = \Omega(n^2)$$

This holds because for palindromes, the machine must compare $\Omega(n)$ bits across the middle of the string, and each bit of information about the prefix x that is compared with z must be stored in the work tape space or carried through state transitions over the distance of y. Therefore, $T(n) \cdot S(n) = \Omega(n^2)$. □

N.2.5 Grok

Proof. We prove the two parts of the theorem separately.

For part (a): We construct a co-nondeterministic Turing machine M recognizing L_{pal} in $O(\log n)$ space and $O(n)$ time. A co-nondeterministic Turing machine accepts an input if and only if every computation path accepts (assuming all paths halt).

On input w of length n, M proceeds as follows:

- Deterministically scan the input to compute the length n, using a binary counter on the work tape. This requires $O(\log n)$ space and $O(n)$ time.

- Nondeterministically guess an integer $i \in \{1, \ldots, \lfloor n/2 \rfloor\}$ by writing its binary representation on the work tape ($O(\log n)$ space).

- Deterministically perform the following on each path: - Return the input head to the left end of the tape. - Advance the input head to position i while counting, and read symbol $\sigma = w_i$ ($O(n)$ time). - Compute $j = n + 1 - i$. - Return the input head to the left end. - Advance to position j and read symbol $\tau = w_j$ ($O(n)$ time). - If $\sigma = \tau$, accept on this path; otherwise, reject.

The total space per path is $O(\log n)$, and the time per path is $O(n)$.

If $w \in L_{\text{pal}}$, then $w_i = w_{n+1-i}$ for all i, so every path accepts, and M accepts. If $w \notin L_{\text{pal}}$, there exists some i with $w_i \neq w_{n+1-i}$, so the path guessing that i rejects, and M rejects. (For odd n, the middle symbol is unchecked but matches itself trivially.)

Thus, M is a co-nondeterministic Turing machine recognizing L_{pal} in $O(\log n)$ space and $O(n)$ time.

For part (b): Suppose, for the sake of contradiction, that there exists a nondeterministic Turing machine M recognizing L_{pal} in time $T(n)$ and space $S(n)$ with $T(n)S(n) = o(n^2)$.

Consider inputs of even length $n = 2m$ with m large (assume $\Sigma = \{0, 1\}$ without loss of generality). For each $u \in \{0,1\}^m$, the string $w = u\,u^R$ is a palindrome of length n, so M has at least one accepting path on w, of length at most $T(n)$ using at most $S(n)$ space.

On such an accepting path for $w = u\,u^R$, M must verify that all m corresponding symbol pairs match. The work tape holds $O(S(n))$ bits of information.

During any contiguous segment where the input head is on the right half, the path can use the information available upon entry (determined by the configuration of size $O(S(n))$ bits) to verify at most $O(S(n))$ pair matches.

To verify all m pairs on an accepting path, there must be $\Omega(m/S(n)) = \Omega(n/S(n))$ distinct visits by the input head to the right half, each bringing fresh information about unmatched pairs from the left half.

Each such visit requires the input head to cross from the left half to the right half and (to fetch the next batch) eventually return to the left half. Traversing from one half to the other requires $\Omega(n)$ time in order to access arbitrary positions.

Thus, each additional batch of verifications costs $\Omega(n)$ time for the round trip. The total time on the accepting path is therefore $\Omega((n/S(n)) \cdot n) = \Omega(n^2/S(n))$.

Hence $T(n) = \Omega(n^2/S(n))$, so $T(n)S(n) = \Omega(n^2)$, contradicting the assumption.

Therefore, any nondeterministic Turing machine recognizing L_{pal} satisfies $T(n)S(n) = \Omega(n^2)$. □

N.3 Problem 3

N.3.1 Problem Statement

Theorem N.3 (12.5)**.** *Consider a model of Turing machines that have two heads on the input tape. Use multiparty communication complexity to prove a time–space tradeoff for such machines.*

N.3.2 GPT

Proof. We are tasked with proving a time-space tradeoff for Turing machines that have two heads on the input tape, using the framework of multiparty communication complexity. Let us denote the two-headed Turing machine as M, which operates on an input string $x \in \Sigma^n$, where Σ is the input alphabet and n is the length of the input. The machine M operates with two heads, H_1 and H_2, each of which can move independently along the input tape.

Step 1: Time and Space Complexity of Two-Headed Turing Machines

Let $t(n)$ denote the time complexity of the machine M on an input of length n, and let $s(n)$ denote the space complexity. Since the machine operates with two heads, the movement of each head is restricted to the tape, and the total space used by the machine is $s(n)$, which is proportional to the number of tape cells accessed by both heads. Thus, the space complexity is bounded by the number of tape cells, which is $s(n) = O(n)$ in the worst case.

The time complexity $t(n)$ depends on the number of steps the machine takes to halt, which can be bounded by $t(n) = O(n^2)$ in the worst case, assuming the heads move independently and the machine is not performing any inefficient operations. Therefore, we have the basic bounds:

$$s(n) = O(n), \quad t(n) = O(n^2).$$

Step 2: Multiparty Communication Complexity

To apply multiparty communication complexity, we need to represent the computation of the two-headed Turing machine as a communication problem between multiple parties. Each head, H_1 and H_2, will correspond to a separate party, and the communication between these parties will simulate the behavior of the Turing machine.

Consider a two-party communication problem where the first party, corresponding to H_1, holds part of the input string $x_1 \in \Sigma^{n/2}$, and the second party, corresponding to H_2, holds the remaining part $x_2 \in \Sigma^{n/2}$. The goal is to compute the output of the Turing machine after both heads have scanned the input string.

The communication complexity of this problem is governed by the number of bits that must be exchanged between the two parties to compute the output. Since the two heads can move independently, the communication complexity can be related to the time complexity of the machine. Specifically, if the machine takes $t(n)$ steps, then the communication complexity must also be at least $t(n)$ bits, since each bit of communication corresponds to a potential action of the machine.

Step 3: Time-Space Tradeoff

Now, consider the tradeoff between time and space for such a machine. From the communication complexity perspective, the amount of time required for the machine to compute its output is at least proportional to the amount of communication that must occur between the two heads. Since the heads can move independently, the total time complexity $t(n)$ is proportional to the number of communication steps, which is bounded by $O(n^2)$.

On the other hand, the space complexity is bounded by $O(n)$, as the tape space is limited to n cells. Therefore, we can conclude that there is a tradeoff between time and space: to reduce the time complexity, the machine must use more space, and conversely, to reduce the space complexity, the machine must take more time.

Conclusion

We have shown that the two-headed Turing machine exhibits a time-space tradeoff. Specifically, the time complexity $t(n)$ is bounded by $O(n^2)$, and the space complexity $s(n)$ is bounded by $O(n)$. This tradeoff is a consequence of the multiparty communication complexity framework applied to the problem,

where the time complexity corresponds to the communication complexity between the two heads, and the space complexity is determined by the tape size.

Thus, we have established the desired time-space tradeoff for two-headed Turing machines. □

N.3.3 Claude

Proof. We prove a time-space tradeoff for two-headed Turing machines using multiparty communication complexity.

Let M be a two-headed Turing machine with heads H_1 and H_2 operating on an input x of length n. Suppose M runs in time T and space S on inputs of length n.

We partition the computation into $r = \lceil T/S \rceil$ epochs, where each epoch consists of S consecutive time steps. For each epoch $i \in \{1, 2, \ldots, r\}$, we define the configuration at the beginning of epoch i by the tuple (q_i, p_1^i, p_2^i, w_i) where q_i is the state, p_1^i and p_2^i are the positions of heads H_1 and H_2 respectively, and w_i is the work tape content (of length at most S).

We establish a connection to three-party communication complexity. Consider three parties: Alice receives the input symbols read by H_1 during the computation, Bob receives the input symbols read by H_2, and Charlie receives the initial configuration. The parties must jointly compute whether M accepts x.

For each epoch i, the computation depends only on: (1) the configuration at the start of the epoch, (2) the input symbols accessed by H_1 during epoch i, and (3) the input symbols accessed by H_2 during epoch i. Since each head can move at most S positions during an epoch of length S, each head accesses at most S input symbols per epoch.

We construct a communication protocol as follows. Charlie sends the initial configuration (requiring $O(S \log n)$ bits). For each epoch $i = 1, \ldots, r$, Alice sends the at most S symbols read by H_1 (requiring $O(S)$ bits), Bob sends the at most S symbols read by H_2 (requiring $O(S)$ bits), and the parties jointly compute the configuration at the end of epoch i (requiring $O(S \log n)$ bits to transmit).

The total communication is bounded by $O(S \log n + r \cdot S) = O(S \log n + (T/S) \cdot S) = O(S \log n + T)$.

By known lower bounds in multiparty communication complexity, for certain explicit functions $f : \{0,1\}^n \to \{0,1\}$, any three-party protocol computing f requires communication $\Omega(n)$.

Therefore, we must have $S \log n + T = \Omega(n)$, which yields the time-space tradeoff:

$$T \cdot S = \Omega(n^2 / \log n).$$

More precisely, if $S = o(n/\log n)$, then $T = \Omega(n^2/(S \log n))$, establishing that $T \cdot S = \Omega(n^2/\log n)$ for any two-headed Turing machine computing functions with high multiparty communication complexity.

This demonstrates that two-headed Turing machines cannot simultaneously achieve sublinear space and linear time for such hard functions, proving the desired time-space tradeoff. □

N.3.4 Gemini

Proof. Let M be a deterministic Turing machine with two heads on a read-only input tape of length n, and a read-write work tape of size $S(n)$. We aim to prove a lower bound on the time $T(n)$ taken by M to recognize the language of palindromes $L_{pal} = \{w \in \{0,1\}^n : w = w^R\}$.

We utilize the framework of multiparty communication complexity in the "Number On the Forehead" (NOF) model. Divide the input w into k contiguous blocks $x_1, x_2, \ldots, x_k$, each of size $m = n/k$. In the k-party communication protocol, player P_i sees all blocks except x_i.

For the two-headed Turing machine, define a "crossing" to occur when both heads are simultaneously located within the same block x_i, or when one head crosses a boundary between blocks. However, to leverage the power of the NOF model, we observe that at any time step t, at least one block x_j contains no input heads, as there are k blocks and only 2 heads.

Let $k = 3$. Divide the input into three equal parts x_1, x_2, x_3. The total number of configurations of the work tape and head positions is given by $C = |Q| \cdot n^2 \cdot |\Gamma|^{S(n)}$, where Q is the state set and Γ is the work tape alphabet. This simplifies to $2^{O(S(n)+\log n)}$.

We partition the execution of M into time intervals. Within any interval where no head crosses a boundary between x_1, x_2, and x_3, the transition of the machine depends only on the work tape state and

the symbols under the two heads. If head 1 is in x_i and head 2 is in x_j, then for $k = 3$, there is always a player P_ℓ ($\ell \neq i, j$) who knows the input symbols under both heads.

The total information exchanged between the three players to simulate M is bounded by the number of times a head crosses a boundary, multiplied by the bits required to describe the configuration, which is $O(S(n) + \log n)$. Let W be the number of boundary crossings. The communication complexity $D(L_{pal})$ is thus $O(W \cdot (S(n) + \log n))$.

From communication complexity theory, it is known that for certain functions f, the k-party NOF complexity is high. For the identity function (inherent in palindrome recognition where x_1 must match x_3^R), the communication required is $\Omega(n/k)$. Setting $k = 3$, we have:

$$W \cdot (S(n) + \log n) = \Omega(n)$$

Since each crossing is a result of moving a head, and the total time $T(n)$ is at least the sum of moves between boundaries, we have $T(n) \geq W \cdot (n/k)$. Substituting W from the communication bound:

$$T(n) \cdot (S(n) + \log n) = \Omega(n^2)$$

This yields the time-space tradeoff $T(n) \cdot S(n) = \Omega(n^2)$ for $S(n) = \Omega(\log n)$, demonstrating that even with two heads, the machine cannot significantly bypass the quadratic bottleneck for palindrome recognition without sufficient space. □

N.3.5 Grok

Proof. We prove that Turing machines with a read-only input tape equipped with two independent heads and a separate read-write work tape admit a language requiring a time-space tradeoff of the form $T(n)S(n) = \Omega(n^{3/2})$.

Let $f : (\{0,1\}^m)^3 \to \{0,1\}$ be an explicit function requiring $\Omega(\sqrt{m})$ bits of communication in the deterministic 3-party number-on-forehead (NOF) communication model. Such functions exist; for concreteness, one may take a suitable variant based on 3-party disjointness known to have this lower bound.

Define the language

$$L = \{x\Delta y\Delta z \mid |x| = |y| = |z| = m,\ f(x, y, z) = 1\},$$

where Δ is a separator symbol not in $\{0, 1\}$, and $n = 3m + 2$.

Assume, for the sake of contradiction, that there exists a two-head Turing machine M recognizing L using time $T(n)$ and space $S(n)$ with $T(n)S(n) = o(n^{3/2})$.

The input tape is divided into three regions of length m each (left: x, middle: y, right: z), separated by the symbols Δ.

A phase of the computation of M is a maximal contiguous sequence of steps during which the set of occupied regions (regions containing at least one head) remains constant.

A phase is long if it consists of at least cm steps, for a sufficiently large constant c allowing the heads to traverse their occupied regions multiple times; otherwise, it is short.

The number of long phases is at most $T(n)/(cm) = O(T(n)/n)$.

We now describe a 3-party NOF protocol for f using $o(\sqrt{m})$ communication.

The three players P_x, P_y, P_z hold x, y, z respectively and see each other's inputs but not their own.

They simulate the execution of M on $x\Delta y\Delta z$.

They maintain a shared current internal configuration C (control state and work tape contents, requiring $O(S(n))$ bits).

For a long phase where exactly two regions are occupied (so one region is unseen), the player corresponding to the unseen region sees both occupied inputs fully.

This player can locally simulate the entire long phase: they know all symbols read by the heads (in the seen regions), compute all transitions, track head positions within the regions, and determine the internal configuration at the end of the phase.

This player then broadcasts the updated internal configuration ($O(S(n))$ bits).

The other players update accordingly.

Phases with fewer than two occupied regions or short phases can be handled with additional bounded communication per phase change or by charging to adjacent long phases in standard fashion; their contribution does not affect the asymptotic bound since the dominant cost arises from the $O(T(n)/n)$ long phases with two occupied regions.

The total communication is thus $O((T(n)/n)S(n)) = o(\sqrt{n})$.

At the end of the simulation, the players determine acceptance based on the final internal configuration.

This correctly computes f, contradicting the $\Omega(\sqrt{m}) = \Omega(\sqrt{n})$ lower bound for the NOF communication complexity of f.

Therefore, no such machine M exists, and the claimed time-space tradeoff holds for the model. □

N.4 Problem 4

N.4.1 Problem Statement

Theorem N.4 (Exercise 12.6)**.** *Let c be a constant and consider the finite language $L_c = \{ww \mid w \in \{0,1\}^c\}$.*

***(1)** There exists a co-nondeterministic automaton with $O(c)$ states that accepts the language L_c.*

***(2)** Any deterministic automaton that accepts the language L_c requires at least 2^c states.*

Furthermore, the transformation from nondeterministic automata with k states to deterministic automata with 2^k states is optimal.

N.4.2 GPT

Proof. **(1)** We are tasked with showing that there exists a co-nondeterministic automaton with $O(c)$ states that accepts the language $L_c = \{ww \mid w \in \{0,1\}^c\}$.

Let us first describe the structure of the language L_c. Any string in L_c consists of two identical halves, each of length c. Therefore, a string in L_c can be viewed as the concatenation of two substrings, both of which are words from the alphabet $\{0,1\}$ of length c. More formally, for $w \in \{0,1\}^c$, the string ww belongs to L_c.

Consider a co-nondeterministic automaton $\mathcal{A}$ that operates as follows: - The automaton first guesses the midpoint of the string, splitting the input string into two halves. - Then, it verifies that the second half is identical to the first half.

To check the second half, the automaton uses a co-nondeterministic move to simultaneously compare each corresponding pair of symbols in the two halves of the string. Since the automaton only needs to verify that the string is of the form ww, and since it can nondeterministically guess the split, it requires at most $O(c)$ states to perform this operation. The automaton has one state for each possible comparison, which is clearly bounded by the length of the word w.

Thus, there exists a co-nondeterministic automaton with $O(c)$ states that accepts the language L_c.

(2) Now, we turn to the second part of the problem, which requires us to show that any deterministic automaton accepting L_c requires at least 2^c states.

To do so, we use the fact that the language L_c contains 2^c distinct strings of length $2c$, each of the form ww for some $w \in \{0,1\}^c$. A deterministic automaton that accepts L_c must distinguish between these 2^c possible strings. Since each string in L_c can be associated with a unique word $w \in \{0,1\}^c$, the deterministic automaton must have a separate state for each possible word w in order to keep track of the first half of the string.

Given that there are 2^c possible binary strings of length c, the deterministic automaton must have at least 2^c states in order to accept all strings in L_c. Therefore, the minimal number of states required for a deterministic automaton to accept L_c is at least 2^c.

Furthermore, the transformation from nondeterministic automata with k states to deterministic automata with 2^k states is well known to be optimal, as the power set construction (subset construction) for converting a nondeterministic automaton to a deterministic one results in an automaton with 2^k states, where k is the number of states in the nondeterministic automaton. This transformation is optimal because

no deterministic automaton can accept the language with fewer than 2^k states, where k is the number of states in the original nondeterministic automaton.

Thus, the transformation from nondeterministic automata with k states to deterministic automata with 2^k states is optimal.

□

N.4.3 Claude

Proof. We prove each part of the theorem in turn.

Part (1): We construct a co-nondeterministic automaton (co-NFA) with $O(c)$ states that accepts L_c.

Recall that a co-NFA accepts a word w if and only if all computational paths reject w. Equivalently, a co-NFA rejects w if and only if there exists at least one accepting path. Thus, to accept L_c, we construct a co-NFA that rejects exactly those words not in L_c.

The co-NFA operates as follows: On input x of length $2c$, it nondeterministically guesses a position $i \in \{1, 2, \ldots, c\}$ where the first half and second half of x differ. The automaton then verifies that $x[i] \neq x[c+i]$.

Formally, the co-NFA has states $\{q_0, q_1, \ldots, q_c, q_{\text{acc}}\}$, where:

- q_0 is the initial state
- From q_0, the automaton nondeterministically transitions to any state q_i for $i \in \{1, \ldots, c\}$, reading the first i symbols
- State q_i counts c more positions, reaching position $c + i$
- At position $c + i$, if $x[i] \neq x[c+i]$, transition to q_{acc}
- q_{acc} is the unique accepting state

This automaton has $c + 2 = O(c)$ states. A word x is rejected by the co-NFA (meaning there exists an accepting path) if and only if $x \notin L_c$, since there exists a position where the two halves differ. Therefore, the co-NFA accepts exactly L_c.

Part (2): We prove that any DFA accepting L_c requires at least 2^c states.

Consider the set $S = \{0, 1\}^c$ of all binary strings of length c. We have $|S| = 2^c$. We will show that for any two distinct strings $u, v \in S$ with $u \neq v$, the strings u and v are distinguishable with respect to L_c.

Two strings u and v are distinguishable if there exists a string z such that exactly one of uz and vz is in L_c.

Let $u, v \in S$ with $u \neq v$. Take $z = u$. Then:

$$uz = uu \in L_c \quad \text{(by definition of } L_c\text{)}$$

However, $vz = vu \notin L_c$ since $v \neq u$, so vu is not of the form ww for any $w \in \{0, 1\}^c$.

Therefore, u and v are distinguishable by the suffix $z = u$.

By the Myhill-Nerode theorem, the number of states in any DFA accepting L_c is at least the number of equivalence classes of the indistinguishability relation, which is at least $|S| = 2^c$.

Optimality of the transformation: We have shown that:

- There exists a co-NFA (equivalently, an NFA for the complement $\overline{L_c}$) with $O(c)$ states
- Any DFA for L_c requires $\Omega(2^c)$ states

Since a co-NFA with k states can be converted to an NFA with k states (for the complement language), and the standard subset construction converts an NFA with k states to a DFA with at most 2^k states, our example shows that this exponential blowup is unavoidable in the worst case. Specifically, with $k = O(c)$ states in the co-NFA, we require $2^c = 2^{\Theta(k)}$ states in the DFA, demonstrating that the transformation is optimal. □

N.4.4 Gemini

Proof. (1) To prove that there exists a co-nondeterministic automaton (co-NFA) with $O(c)$ states that accepts L_c, we equivalently show that there exists a nondeterministic finite automaton (NFA) M' with $O(c)$ states that accepts the complement language $\overline{L_c} = \{x \in \{0,1\}^{2c} \mid x \neq ww \text{ for any } w \in \{0,1\}^c\} \cup \{x \in \{0,1\}^* \mid |x| \neq 2c\}$.

An NFA can recognize $\overline{L_c}$ by nondeterministically branching to check one of two conditions: First, it checks if the input length is not $2c$. This requires $2c+2$ states to count up to $2c$ and an extra state to trap any string longer than $2c$. Second, it checks if there exists some index $i \in \{1, \ldots, c\}$ such that the i-th bit of the input differs from the $(c+i)$-th bit. For a fixed i, the NFA uses i states to reach the i-th bit, stores its value (0 or 1) in the state, uses another $c-1$ states to reach the $(c+i)$-th bit, and then checks for inequality.

By nondeterministically choosing the index $i \in \{1, \ldots, c\}$, the NFA requires $O(c)$ states to track the position and the bit value. Specifically, the number of states is roughly $\sum_{i=1}^{c} O(1)$ if we reuse the counting structure or $O(c)$ states using a standard construction for $x_i \neq x_{i+c}$. Since L_c is recognized by a co-NFA if $\overline{L_c}$ is recognized by an NFA of the same size, we conclude L_c has a co-NFA of size $O(c)$.

(2) To prove that any deterministic finite automaton (DFA) M accepting L_c requires at least 2^c states, we employ the Myhill-Nerode theorem. We identify a set of prefixes that are pairwise inequivalent under the Myhill-Nerode relation $\sim_{L_c}$.

Consider the set of strings $S = \{w \mid w \in \{0,1\}^c\}$. Note that $|S| = 2^c$. Let $u, v \in S$ be two distinct strings such that $u \neq v$. To show $u \not\sim_{L_c} v$, we must find a suffix $z \in \{0,1\}^*$ such that exactly one of uz or vz is in L_c.

Let $z = u$. By the definition of L_c, the concatenation uu consists of a string u followed by itself, where $|u| = c$. Thus $uu \in L_c$. Now consider the string vu. Since $|v| = c$ and $|u| = c$, the string vu has length $2c$. For vu to be in L_c, it must be of the form ww for some $w \in \{0,1\}^c$. This implies $v = w$ and $u = w$, which leads to $u = v$. However, we assumed $u \neq v$, therefore $vu \notin L_c$.

Because $u \in L_c$ and $vu \notin L_c$, the strings u and v are in different equivalence classes of $\sim_{L_c}$. Since there are 2^c such strings in S, the Myhill-Nerode relation has at least 2^c distinct equivalence classes. Therefore, any DFA accepting L_c must have at least 2^c states.

The existence of a language L_c recognizable by an NFA (or co-NFA) with $k = O(c)$ states that requires a DFA with $2^c = 2^{\Omega(k)}$ states demonstrates that the powerset construction, which transforms a k-state NFA into a 2^k-state DFA, is asymptotically optimal in the worst case. □

N.4.5 Grok

Proof. We prove each part of the theorem in turn.

For part (1), we construct a co-nondeterministic finite automaton A (also known as a universal nondeterministic finite automaton, where a string is accepted if and only if every possible computation path accepts) that recognizes exactly the language L_c using $O(c)$ states.

The alphabet is $\Sigma = \{0,1\}$. The state set Q consists of: - Main states $p_0, p_1, \ldots, p_{2c}$, - Checking states skip_m^b for $b \in \{0,1\}$ and $m = 0, 1, \ldots, c-1$, - Sink states accept and reject.

This gives a total of $2c + 1 + 2c + 2 = O(c)$ states. The start state is p_0. The accepting states are p_{2c} and accept. (A path accepts if it ends in an accepting state after reading the entire input; paths with no transition on a symbol or ending in reject reject.)

The transition function $\delta : Q \times \Sigma \to 2^Q$ (nonempty sets where defined) is as follows, where $\mathrm{bit}(a) = 0$ if $a = 0$ and 1 if $a = 1$:

- For $k = 0$ to $c-1$ and each $a \in \Sigma$, let $b = \mathrm{bit}(a)$:

$$\delta(p_k, a) = \{p_{k+1}, \mathrm{skip}_0^b\}.$$

- For $k = c$ to $2c-1$ and each $a \in \Sigma$:

$$\delta(p_k, a) = \{p_{k+1}\}.$$

- For $m = 0$ to $c - 2$, $b \in \{0, 1\}$, and each $a \in \Sigma$:

$$\delta(\text{skip}_m^b, a) = \{\text{skip}_{m+1}^b\}.$$

- For $m = c - 1$, $b \in \{0, 1\}$, and each $a \in \Sigma$, let $b' = \text{bit}(a)$:

$$\delta(\text{skip}_{c-1}^b, a) = \begin{cases} \{\text{accept}\} & \text{if } b' = b, \\ \{\text{reject}\} & \text{otherwise.} \end{cases}$$

- For each $a \in \Sigma$:

$$\delta(\text{accept}, a) = \{\text{accept}\}, \quad \delta(\text{reject}, a) = \{\text{reject}\}.$$

- No transitions are defined from p_{2c} or in other cases (implicit reject for paths lacking transitions).

On an input s of length exactly $2c$ with $s = ww$ for some $w \in \{0, 1\}^c$, the computation universally branches during the first c steps, spawning exactly one checking path at each position $j = 1$ to c that remembers w_j and checks the symbol at position $c + j$. The main path reaches p_{2c}. Each checking path reaches accept since all pairs match. Thus, all paths accept, so A accepts.

If $s = uv$ with $|u| = |v| = c$ but $u \neq v$, then there exists j with $u_j \neq v_j$, so the checking path spawned at position j reaches reject. Thus, not all paths accept, so A rejects.

For inputs of length not equal to $2c$, some paths lack transitions before completing or the main path fails to reach p_{2c}, so not all paths accept (or no complete paths to accepting states), and A rejects.

Thus, A recognizes exactly L_c with $O(c)$ states.

For part (2), consider the Myhill–Nerode equivalence relation on Σ^*. For distinct $u, v \in \{0, 1\}^c$ with $u \neq v$, let $z = u$. Then $uz = uu \in L_c$, but $vz = vu \notin L_c$ (since if $vu = ww'$ for some $|w| = |w'| = c$, then $v = w$ and $u = w'$, so $v = u$, a contradiction). Thus, $u \not\equiv v$. There are 2^c such distinct strings of length c in different equivalence classes. Therefore, any deterministic finite automaton recognizing L_c requires at least 2^c states.

Furthermore, the standard subset construction transforming a nondeterministic finite automaton with k states into an equivalent deterministic finite automaton may produce up to 2^k states, and this exponential blowup is optimal: there exist languages recognized by nondeterministic finite automata with k states but requiring 2^k states for any deterministic finite automaton. □

N.5 Problem 5

N.5.1 Problem Statement

Theorem N.5. *Let $f : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}$ be a function. Let M be a one-tape non-deterministic Turing machine with the same properties as in Lemma 12.7. That is, the machine M accepts all inputs in $\{x0^n y \mid |x| = |y| = n, f(x, y) = 1\}$, rejects all inputs in $\{x0^n y \mid |x| = |y| = n, f(x, y) = 0\}$, and runs in time $T(n)$ (where the time of a nondeterministic Turing machine is defined as the maximum over all inputs in $\{0, 1\}^n$ and all possible nondeterministic choices). Then, $N^1(f) = O((T(n)/n) + \log n)$. Conclude that such a nondeterministic Turing machine for the language of palindromes requires $\Omega(n^2)$ times.*

N.5.2 GPT

Proof. We are given a function $f : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}$ and a one-tape non-deterministic Turing machine M that accepts all inputs in the language $L_f = \{x0^n y \mid |x| = |y| = n, f(x, y) = 1\}$, rejects all inputs in the language $\overline{L_f} = \{x0^n y \mid |x| = |y| = n, f(x, y) = 0\}$, and runs in time $T(n)$. The goal is to show that $N^1(f) = O\left(\frac{T(n)}{n} + \log n\right)$, and subsequently conclude that a nondeterministic Turing machine for the palindrome language requires $\Omega(n^2)$ time.

Step 1: Define $N^1(f)$

The term $N^1(f)$ represents the time complexity of a one-tape nondeterministic Turing machine that decides the function f in the worst-case scenario. This complexity is defined as the maximum time taken by the machine over all inputs and all nondeterministic choices. Therefore, we are tasked with showing that the time complexity of this machine is at most $O\left(\frac{T(n)}{n} + \log n\right)$.

Step 2: Analyze the time complexity of M

The machine M is a one-tape nondeterministic Turing machine that operates in time $T(n)$. This implies that for any input of size n, the machine will make at most $T(n)$ steps, where $T(n)$ is a function of the input size n and represents the worst-case time complexity for nondeterministic computation.

Step 3: Relating $T(n)$ to $N^1(f)$

To evaluate $N^1(f)$, we consider how the machine M operates on the input. Specifically, the machine reads and processes two strings, x and y, each of length n, with an additional separator 0^n. For a nondeterministic Turing machine, the computation splits into several branches based on its nondeterministic choices. At each step, the machine makes a decision about how to continue processing the input.

Since M runs in time $T(n)$, the number of steps it takes to process the input depends on how the machine nondeterministically chooses to process the input. The time complexity of the nondeterministic machine can be described as $O\left(T(n)\right)$, and we must account for the structure of the input.

Step 4: Factor in the input structure

The input to the machine consists of two strings, x and y, each of length n. Since the machine processes these strings together with the separator 0^n, the time complexity will also depend on how the machine accesses and processes these components. The processing of each string involves reading n symbols, and the time complexity is affected by how these symbols are accessed sequentially.

Given that the machine operates on two strings of length n, the time complexity is dominated by $T(n)$, which is the time required to nondeterministically decide the function f. However, since f is defined on pairs of strings, the total time complexity must also account for the time required to process each string independently.

Step 5: Estimating the overall time complexity

The overall time complexity $N^1(f)$ can be bounded by considering the number of steps required to process each string. Each string of length n takes at most $O(n)$ steps to process. Thus, the time complexity of the machine is at most $O\left(\frac{T(n)}{n} + \log n\right)$, where the $\frac{T(n)}{n}$ term reflects the processing of the strings x and y, and the $\log n$ term accounts for the nondeterministic choices made by the machine.

Step 6: Applying to the palindrome language

Now, we apply this analysis to the case where the function f corresponds to the language of palindromes. A palindrome is a string that reads the same forward and backward, and recognizing palindromes requires checking that the first half of the string matches the second half in reverse order.

For a nondeterministic Turing machine deciding the palindrome language, we observe that the machine must check pairs of characters, which requires at least n^2 time in the worst case. Therefore, we conclude that any nondeterministic Turing machine that decides the palindrome language must require $\Omega(n^2)$ time.

Thus, we have shown that $N^1(f) = O\left(\frac{T(n)}{n} + \log n\right)$ and that a nondeterministic Turing machine for the palindrome language requires $\Omega(n^2)$ time.

□

N.5.3 Claude

Proof. We prove the theorem in two parts: first establishing the upper bound on $N^1(f)$, then applying it to palindromes.

Part 1: Upper bound on $N^1(f)$.

Let M be a one-tape nondeterministic Turing machine that decides f in time $T(n)$ as specified. We construct a nondeterministic communication protocol for f with communication complexity $O((T(n)/n) + \log n)$.

Given input (x, y) where $|x| = |y| = n$, Alice holds x and Bob holds y. The protocol proceeds as follows:

Alice nondeterministically guesses a computation path of M on input $x0^n y$. Since M runs in time $T(n)$, the head position at any time t is at most $T(n)$ from the initial position.

We partition the computation into phases based on the head position. Define a *crossing sequence* at position i as the sequence of states in which the head crosses position i during the computation. Since M has a constant number of states c and runs in time $T(n)$, each crossing sequence has length at most $T(n)$.

Consider the boundary position between Alice's input x and the separator 0^n. Alice nondeterministically guesses the crossing sequence σ at this boundary. The length of this crossing sequence is at most $T(n)$, so Alice can communicate it using $O(T(n) \log c) = O(T(n))$ bits.

However, we must be more careful. The head may cross the boundary at most $T(n)$ times total, but we can compress this information. Alice sends:

(i) The crossing sequence at position n (between x and 0^n), which requires $O(T(n))$ bits.

(ii) The position where the head is at time $T(n)$, which requires $O(\log T(n))$ bits.

Since the tape contains $x0^n y$ and has length $3n$, and the head moves at most $T(n)$ positions, we can divide the tape into $O(T(n)/n)$ blocks of size n. The number of times the head crosses between blocks is at most $O(T(n)/n)$.

For each block boundary, Alice and Bob exchange the crossing sequence. Each crossing sequence at a block boundary requires $O(n)$ bits to describe (since there are at most n crossings per block boundary in the worst case, and each crossing involves a constant-size state).

More precisely, Alice sends the crossing sequence at the boundary between her portion and Bob's portion. This crossing sequence has length at most $T(n)/n$ (amortized), requiring $O((T(n)/n) \cdot \log c) = O(T(n)/n)$ bits.

Additionally, Alice must send $O(\log n)$ bits to specify the exact position of certain key configurations.

Bob can then verify whether his portion y is consistent with the guessed computation path and the crossing sequence. Bob accepts if and only if there exists a valid accepting computation of M consistent with the crossing sequence.

Therefore, $N^1(f) = O((T(n)/n) + \log n)$.

Part 2: Lower bound for palindromes.

Let $\text{PAL}(x, y) = 1$ if and only if xy is a palindrome, i.e., $x_i = y_{n+1-i}$ for all $i \in [n]$.

It is known that $N^1(\text{PAL}) = \Omega(n)$. This follows from the fact that Alice must communicate essentially all of x to Bob (or vice versa) for Bob to verify the palindrome property, even nondeterministically.

By the contrapositive of Part 1, if $N^1(\text{PAL}) = \Omega(n)$, then:

$$n = O((T(n)/n) + \log n)$$

This implies:

$$n - \log n = O(T(n)/n)$$

Therefore:

$$\Omega(n) = O(T(n)/n)$$

which gives us $T(n) = \Omega(n^2)$.

Thus, any one-tape nondeterministic Turing machine deciding palindromes requires $\Omega(n^2)$ time. □

N.5.4 Gemini

<think>**Defining the Problem's Scope**

I'm focused on grasping the specific parameters of this problem. My role is clear: construct a rigorous proof. The initial step involves a deep dive into the theorem statement, meticulously extracting key elements such as the communication complexity component ($N^1(f)$) and the Turing machine specifications. This phase is crucial for building a solid foundation.

Revising Proof Structure

I'm now restructuring the proof strategy. The approach centers on connecting the NTM's runtime to communication complexity through crossing sequences. Specifically, I'm focusing on how Alice and Bob can collaborate to simulate the NTM's behavior using communication. The initial plan involves defining the input structure, which is crucial for establishing the communication model.

Synthesizing Crossing Sequences

I am now focusing on the application of the crossing sequence technique in the context of nondeterministic communication complexity. I will build upon the previous analysis of crossing sequences,

incorporating nondeterministic elements. My aim is to define how Alice and Bob can efficiently simulate the NTM's behaviour by constructing and verifying these sequences. The key is in how Alice and Bob agree on the crossing information.

Elaborating the Communication Protocol

I'm now refining the communication protocol details. It's becoming clearer how Alice and Bob will simulate the NTM. Alice and Bob will guess the crossing sequence at the boundary i. This is crucial. Alice and Bob will both compute a witness and must agree on the crossing index and crossing sequence. A crucial component is the verification step: they will check the correctness of their respective computations based on the shared sequence and index.

Finalizing Proof Structure

I'm now putting the finishing touches on the proof outline. I'm focusing on incorporating the specifics of the palindrome language and its relation to the communication complexity bound. The central challenge now involves rigorously connecting the communication protocol's cost to the NTM's runtime, paying close attention to the implications for both the equality function. The final step is to refine the LaTeX code.

Implementing the Proof's Protocol

I'm now focusing on a more formal presentation of the communication protocol for Alice and Bob. The core of the proof now hinges on their ability to convincingly agree on the crossing sequence information, especially where the palindrome split occurs. I'm exploring the specific information exchanged and how it relates to the runtime bound. The next step is a clear demonstration of the lower bound for palindromes.

Formulating Proof Steps

I'm now structuring the formal proof steps. I'm focusing on clarifying how Alice and Bob use the crossing sequence approach for the palindrome problem. The goal is to define the communication protocol, particularly how they agree on the "crossing" information and utilize it for verifying the palindromic property of the input. I will need to use a formal notation to clearly describe this.

Defining Acceptance Conditions

I'm now clarifying the acceptance conditions of the NTM and how it interacts with the function, f. The proof leverages these properties to constrain the NTM behavior. My focus now is on precisely capturing the relation between the machine's time, the input format, and the constraints of the acceptance criteria to accurately model the communication protocol.

Constructing the Core Argument

I'm now focusing on crafting the central argument, connecting the palindrome language to the given communication complexity bound. My approach uses the crossing sequences and protocol design, carefully adapting them to the nondeterministic model. The main challenge now is to link the NTM's runtime and the communication cost for the equality function.

Verifying Protocol Steps

I'm now consolidating the steps in the proof, ensuring the communication protocol's soundness for the chosen input format. The goal is to verify the claimed relationship between the NTM's runtime and the nondeterministic communication complexity. The main task is making the palindrome property integral to the proof's logic.

</think>

Proof. Let M be the one-tape nondeterministic Turing machine described in the theorem, which decides the language $L_f = \{x0^n y \mid x, y \in \{0,1\}^n, f(x,y) = 1\}$ in time $T(n)$. We establish the upper bound on the nondeterministic communication complexity $N^1(f)$ by constructing a protocol based on the computation of M.

Consider an input pair (x, y) such that $f(x, y) = 1$. Since M accepts $x0^n y$, there exists an accepting computation path $\mathcal{P}$ of length at most $T(n)$. The input on the tape consists of x in cells 1 to n, followed by 0^n in cells $n+1$ to $2n$, followed by y in cells $2n+1$ to $3n$. For any integer i such that $0 \leq i \leq n$, let b_i denote the boundary between cell $n+i$ and cell $n+i+1$. A crossing sequence C_i at boundary b_i is defined as the sequence of states $q_1, q_2, \ldots, q_m$ the machine is in when the read/write head crosses b_i, along with the direction of the movement.

The sum of the lengths of the crossing sequences across all boundaries within the 0^n region is bounded by the total running time, as each step of the machine crosses at most one boundary. Thus, for any accepting path $\mathcal{P}$, we have $\sum_{i=0}^{n} |C_i| \leq T(n)$. By the averaging argument (pigeonhole principle), there exists an index $k \in \{0, \ldots, n\}$ such that the length of the crossing sequence satisfies $|C_k| \leq \frac{T(n)}{n+1} < \frac{T(n)}{n}$.

We define a nondeterministic communication protocol for Alice and Bob. Alice holds x and Bob holds y. They nondeterministically guess an index $k \in \{0, \ldots, n\}$ and a crossing sequence $C = (s_1, \ldots, s_m)$ representing the states and directions at boundary b_k. The cost to communicate this guess is $O(m + \log n)$, where $m = |C|$, assuming the set of states Q is constant.

Alice verifies that there exists a partial computation of M on the tape portion containing x and 0^k (cells 1 to $n + k$) that is consistent with the crossing sequence C at the right boundary and the initial configuration. Bob verifies that there exists a partial computation on the tape portion containing 0^{n-k} and y (cells $n + k + 1$ to $3n$) consistent with C at the left boundary and ending in an accepting state. If both verifications succeed, they output 1. Since M is nondeterministic, $N^1(f)$ is the minimum cost of such a witness for an accepting input. By the averaging argument above, there exists a witness with length $m \leq T(n)/n$. Therefore, the nondeterministic communication complexity is bounded by:

$$N^1(f) = O\left(\frac{T(n)}{n} + \log n\right).$$

To conclude the lower bound for palindromes, let $L_{pal} = \{w \in \{0,1\}^* \mid w = w^R\}$. We restrict our attention to inputs of the form $w = x0^n y$ where $|x| = |y| = n$. Note that $x0^n y \in L_{pal}$ if and only if $x0^n y = (x0^n y)^R = y^R 0^n x^R$, which implies $x = y^R$. Let $f(x, y) = 1$ if $x = y^R$ and 0 otherwise. This function is equivalent to the Equality function $EQ(x, z)$ where $z = y^R$.

It is a standard result in communication complexity that $N^1(EQ) \geq n$. Specifically, for any two distinct inputs $x \neq x'$, the sets of y such that $f(x, y) = 1$ and $f(x', y) = 1$ are disjoint, implying the fooling set size is 2^n, or alternatively, to nondeterministically verify $x = z$, one must reveal x entirely. Substituting this lower bound into our derived upper bound:

$$n \leq O\left(\frac{T(n)}{n} + \log n\right).$$

For sufficiently large n, the $\log n$ term is negligible compared to n, implying $n \leq c\frac{T(n)}{n}$ for some constant c. Thus, $T(n) \geq \frac{1}{c}n^2$. We conclude that any one-tape nondeterministic Turing machine recognizing palindromes requires time $T(n) = \Omega(n^2)$. □

N.5.5 Grok

Proof. We first prove that $N^1(f) = O(T(n)/n + \log n)$, where $N^1(f)$ denotes the nondeterministic communication complexity of f (the minimum length of a proof π such that, if $f(x, y) = 1$ there exists π for which both parties accept given their input and π, and if $f(x, y) = 0$ then for every π at least one party rejects).

Consider inputs of the form $x0^n y$ with $|x| = |y| = n$ placed on the tape as positions 1 to n: symbols of x, positions $n + 1$ to $2n$: 0's, positions $2n + 1$ to $3n$: symbols of y, and blanks elsewhere. The head starts at position 1.

Fix a boundary in the middle of the separator, say between positions $n + \lfloor n/2 \rfloor$ and $n + \lfloor n/2 \rfloor + 1$. Traversing from one side of this boundary to the other requires at least $\lfloor n/2 \rfloor$ steps. Thus, in any computation path of length at most $T(n)$, the head crosses this boundary at most $O(T(n)/n)$ times.

For an accepting input ($f(x, y) = 1$), there exists an accepting path of length at most $T(n)$. Let π be the crossing sequence for this path at the fixed boundary: first the number $m = O(T(n)/n)$ of crossings (encoded in $O(\log n)$ bits, assuming $T(n) \leq n^c$ for some constant c, as is standard), followed by the sequence of m pairs (state, direction) when crossing the boundary, each pair requiring $O(1)$ bits. Thus, $|\pi| = O(T(n)/n + \log n)$.

The parties proceed as follows with this π:

- Alice (holding x) considers the left half-tape (positions $\leq n + \lfloor n/2 \rfloor$, with input x followed by $\lfloor n/2 \rfloor$ 0's, blanks elsewhere) and checks whether there exist nondeterministic choices such that some computation path starting from the initial configuration produces exactly the subsequence of crossings in the left-to-right directions (or appropriately depending on direction convention) matching π, and is consistent with the overall path until halt or end of sequence.

- Bob (holding y) similarly checks the right half-tape (positions $\geq n + \lfloor n/2 \rfloor + 1$, with $\lceil n/2 \rceil$ 0's followed by y, blanks elsewhere) for the right-to-left crossings (or matching subsequence).

- Additionally, the parties verify consistency of the glued path: the number and timing of crossings match, the final state is accepting if the path halts in an accepting state, and no inconsistency arises in head movement or halting.

If $f(x, y) = 1$, the prover sends the crossing sequence π of the existing accepting path; both parties will find consistent local nondeterministic computations realizing their side of π, and the glued path accepts.

If $f(x, y) = 0$, no accepting path exists, so no π allows both parties to find consistent local realizations; at least one rejects.

Thus, the protocol computes f with proof length $O(T(n)/n + \log n)$, so $N^1(f) = O(T(n)/n + \log n)$.

For the language of palindromes L_{pal}, consider instances $w = u0^n v$ with $|u| = |v| = n$; $w \in L_{\text{pal}}$ if and only if $v = u^R$. Define $f(u, v) = 1$ iff $v = u^R$. This f is equivalent (up to relabeling positions) to the equality function, whose communication matrix has 2^n 1's on a permuted diagonal. Each 1-monochromatic rectangle covers at most one 1, so at least 2^n such rectangles are needed to cover the 1's. Thus, $N^1(f) = \Omega(n)$.

If there exists a nondeterministic one-tape TM recognizing L_{pal} in time $T(n)$, it also correctly decides these padded instances, yielding $N^1(f) = O(T(n)/n + \log n) = \Omega(n)$. Hence, $T(n)/n = \Omega(n - \log n)$, so $T(n) = \Omega(n^2)$. □

N.6 Problem 6

N.6.1 Problem Statement

Problem N.6 (Open Problem 12.10)**.** *Prove any super-linear lower bound for computing an explicit function on two-tape Turing machines.*

N.6.2 GPT

Proof. The displayed statement is explicitly presented as an open problem: it asks to "Prove any super-linear lower bound for computing an explicit function on two-tape Turing machines." By definition, an open problem is a mathematical assertion (or family of assertions) for which no complete, accepted proof (or disproof) is currently known within the relevant standard formal framework (e.g. ZFC or other commonly adopted axiomatic systems for mathematics and theoretical computer science).

Assume, for the sake of contradiction, that a complete and rigorous proof of the requested claim is presently available as a generally accepted mathematical result. Then the problem would no longer be open; it would be resolved by that proof, contradicting the designation and status of the statement as an open problem. Hence, under the premise that the statement is indeed an open problem in the usual mathematical sense, there is no currently known complete and rigorous proof of the requested super-linear lower bound for two-tape Turing machines computing an explicit function.

Therefore, a complete formal proof of the requested claim cannot be provided here, because the claim is not presently established as a theorem. □

N.6.3 Claude

Proof. This problem remains open. We present the current state of knowledge and the barriers to proving such a result.

Let $f : \{0, 1\}^* \to \{0, 1\}$ be an explicit function (computable in polynomial time). We seek to prove that any two-tape Turing machine computing f requires time $T(n) = \omega(n)$ for inputs of length n.

Current Best Lower Bound: The strongest known lower bound for two-tape Turing machines is $T(n) = \Omega(n \log n)$ for specific functions such as palindrome recognition, due to the crossing sequence

technique. However, no super-linear lower bound $T(n) = \omega(n)$ is known for an explicit function that applies universally to all two-tape machines.

Barrier Analysis: We identify the fundamental obstacles:

(1) Crossing Sequence Limitation: For a two-tape Turing machine with input tape and work tape, let C_i denote the crossing sequence at position i on the work tape. By the crossing sequence argument, if $|C_i| \leq k$ for all positions, then the machine runs in time $O(kn)$. To prove $\omega(n)$ lower bounds, we must show that for any machine computing f, there exists a position where $|C_i| = \omega(1)$.

(2) Compression Argument: Suppose toward contradiction that f can be computed in time cn for some constant c. Then for inputs of length n, the machine makes at most cn steps. The configuration space has size bounded by $|Q| \cdot n \cdot n \cdot |\Gamma|^{cn}$ where Q is the state set and Γ is the tape alphabet. For this to distinguish 2^n inputs, we require $|\Gamma|^{cn} \geq 2^n/(\text{poly}(n))$, giving $|\Gamma| \geq 2^{1/c-o(1)}$. However, this does not yield a contradiction for fixed Γ.

(3) Information Transfer: Any proof must account for the fact that two tapes allow parallel information storage and retrieval. The work tape can be organized to minimize head movement through careful encoding schemes.

Conclusion: Without additional structural assumptions on the function f or restrictions on the machine model (such as oblivious machines or single-tape machines), the techniques currently available in computational complexity theory are insufficient to prove the required super-linear lower bound. The problem requires fundamentally new ideas beyond crossing sequences, communication complexity, and known diagonalization techniques.

Therefore, this problem remains open, and no complete proof can be provided at this time. □

N.6.4 Gemini

Proof. Let M be a deterministic two-tape Turing machine (one read-only input tape and one read-write work tape). We consider the language $L = \{w\#w : w \in \{0,1\}^*\}$ of equality, which is a variation of the palindrome language. We aim to show that $T(n) = \Omega(n^2)$ for this language, which constitutes a super-linear lower bound.

For a fixed input length n, consider inputs of the form $x\#y$ where $|x| = |y| = m$ and $n = 2m + 1$. Let $X = \{0,1\}^m$ be the set of possible strings for x. We analyze the crossing sequence at the boundary marked by the symbol $\#$.

A configuration of the work tape consists of the state of the finite control, the position of the work tape head, and the contents of the work tape. Let s denote the space used by the work tape. The number of possible configurations is given by $C(s) = |Q| \cdot s \cdot |\Gamma|^s$, where Q is the set of states and Γ is the work tape alphabet.

Suppose $T(n) = o(n^2)$. We use the method of crossing sequences. For each $i \in \{1, \ldots, m\}$, let S_i be the set of all crossing sequences at the boundary between the i-th and $(i+1)$-th cell of the input tape. However, for a two-tape machine, a crossing sequence at an input head position i records the state and work tape status every time the input head crosses the boundary i.

Let T be the total time. The sum of the lengths of all crossing sequences across all input tape positions is exactly T. By an averaging argument, there must exist some region of the input tape where the crossing sequences are short. Specifically, for the language L, any information about x must be communicated to the part of the tape processing y.

Let $x, x' \in \{0,1\}^m$ with $x \neq x'$. Let $C(x)$ be the crossing sequence generated by M at the boundary of the $\#$ symbol on input $x\#x$. If $C(x) = C(x')$, then by the crossing sequence cut-and-paste lemma, the machine M must also accept the input $x\#x'$. But $x\#x' \notin L$ because $x \neq x'$. Thus, for all 2^m distinct strings in X, the crossing sequences at the $\#$ boundary must be distinct.

Let d be the length of a crossing sequence. The number of distinct crossing sequences of length at most d is approximately k^d for some constant k related to the machine's configuration space. To distinguish 2^m strings, we require:

$$k^d \geq 2^m \implies d = \Omega(m)$$

Since $m = \Theta(n)$, the crossing sequence at the middle must have length $\Omega(n)$. By considering multiple

such boundaries within the first m bits, if we assume the work tape space is small, the machine must repeatedly pass the input head back and forth.

More formally, if the work tape space $S(n)$ is $o(n)$, we can find $\Omega(n)$ positions i on the input tape such that the crossing sequence at i has length L_i. The total time $T(n) = \sum L_i$. For the language L, the information-theoretic requirement for distinct crossing sequences across a swath of the input tape yields:

$$T(n) \geq \sum_{i=n/4}^{3n/4} \Omega(n) = \Omega(n^2)$$

Thus, any two-tape Turing machine recognizing the equality language (or palindromes) requires quadratic time, providing a super-linear lower bound $\Omega(n^2)$ for an explicit function. □

N.6.5 Grok

Proof. This is a well-known open problem in computational complexity theory. As of January 2026, no super-linear time lower bound $T(n) = \omega(n)$ is known for any explicit (natural, non-diagonalization-based) language or function computed by deterministic two-tape Turing machines.

Hierarchy theorems establish the existence of languages requiring super-linear time on two-tape Turing machines via diagonalization, but these languages are not explicit.

For concrete problems such as palindrome recognition, matrix transposition, sorting, and many others in P, linear-time algorithms exist on two-tape (or constant-tape) Turing machines.

Extensive efforts over decades have yielded super-linear lower bounds only for restricted models (e.g., one-tape, off-line, or branching programs), but not for the standard multi-tape model with a constant number of read-write tapes.

Thus, proving any super-linear lower bound for an explicit function on two-tape Turing machines remains unresolved. □

N.7 Problem 7

N.7.1 Problem Statement

Theorem N.7 (Exercise 12.11)**.** *Let $f : \{0,1\}^m \to \{0,1\}$ be any function. Show that there is a branching program for f of length m and width 2^m and branching program of width 3 and length $O(m2^m)$.*

N.7.2 GPT

Proof. Let $f : \{0,1\}^m \to \{0,1\}$ be an arbitrary function. We are tasked with showing that there exists a branching program for f of length m and width 2^m, as well as a branching program of width 3 and length $O(m2^m)$.

Branching Program of Length m and Width 2^m Consider a branching program where each of the m layers corresponds to one of the m input variables. In each layer, the program branches based on the value of a particular input variable. Since the domain of f is $\{0,1\}^m$, each input is an m-tuple of bits, and at each stage, the program will check the value of one of these m bits.

The program will have the following structure: - The input consists of an m-bit string $x = (x_1, x_2, \ldots, x_m)$, where each $x_i \in \{0,1\}$ for $1 \leq i \leq m$. - At the i-th layer of the program, the program will test the value of the i-th bit x_i. If $x_i = 0$, the program follows one branch; if $x_i = 1$, the program follows another branch. - Each possible sequence of decisions determines a path through the branching program, and since each of the m bits can independently take two values, there are exactly 2^m possible paths.

The length of this branching program is m since there are m layers, and the width is 2^m since there are 2^m possible paths corresponding to all possible input strings. Therefore, this is a valid branching program for f of length m and width 2^m.

Branching Program of Width 3 and Length $O(m2^m)$ Next, we construct a branching program of width 3 and length $O(m2^m)$.

We will use a technique known as a "multi-dimensional" branching program. At each layer, the program will test multiple bits simultaneously, grouping the bits into blocks of size 2. Specifically, we divide the m

input bits into $\lceil m/2 \rceil$ blocks, each consisting of two bits. Thus, the number of blocks is at most $m/2$, and each block corresponds to one decision point in the branching program.

For each pair of bits, the program will branch based on whether the pair is $(0,0)$, $(0,1)$, $(1,0)$, or $(1,1)$. This means that at each layer, there are 4 possible decisions to be made, but we limit the branching factor to 3 by using a ternary decision at each stage: - The program will branch according to whether the pair of bits is $(0,0)$, $(0,1)$, or $(1,1)$. In the case of $(1,0)$, we can handle it as an additional "default" branch, effectively treating this case separately and routing it to one of the existing branches.

Each decision point involves checking two bits, and therefore the width of the program at each layer is 3. Since there are at most $m/2$ layers (one for each block), the total number of layers in the branching program is $O(m)$. Furthermore, the total number of possible paths is at most 2^m, as each input string corresponds to one path, and each path is determined by the choices made at each layer.

Therefore, the total length of this branching program is $O(m2^m)$, and the width is 3. This completes the construction of a branching program for f with the desired properties.

Thus, we have shown that there exists a branching program for f of length m and width 2^m, and a branching program of width 3 and length $O(m2^m)$. □

N.7.3 Claude

Proof. We prove both constructions separately.

Part 1: Branching program of length m and width 2^m.

We construct a branching program that computes f by reading the input bits $x_1, x_2, \ldots, x_m$ in order. At each layer i (for $i = 0, 1, \ldots, m$), we maintain all possible partial assignments to the first i bits.

Define the branching program as follows. Layer i contains 2^i nodes, one for each possible assignment to $(x_1, \ldots, x_i)$. Specifically, for each string $\alpha \in \{0,1\}^i$, there is a node v_α at layer i.

For $i < m$, from node v_α at layer i, we have two outgoing edges:

- An edge labeled $x_{i+1} = 0$ going to node $v_{\alpha 0}$ at layer $i+1$
- An edge labeled $x_{i+1} = 1$ going to node $v_{\alpha 1}$ at layer $i+1$

The start node is the unique node at layer 0 (corresponding to the empty string). At layer m, for each node v_α where $\alpha \in \{0,1\}^m$, we assign the output value $f(\alpha)$.

This branching program has length m (since there are m layers of edges) and width 2^m (since layer m has 2^m nodes). On input $x \in \{0,1\}^m$, the computation path follows the unique sequence of nodes corresponding to the prefix of x, eventually reaching node v_x at layer m, which outputs $f(x)$.

Part 2: Branching program of width 3 and length $O(m2^m)$.

We construct a branching program with three types of nodes: a start node s, an accept node t_1, and a reject node t_0.

The idea is to enumerate all strings in $\{0,1\}^m$ in lexicographic order and check each one sequentially. For each string $\alpha \in \{0,1\}^m$, we create a gadget of length m that checks whether the input x equals α. If $x = \alpha$ and $f(\alpha) = 1$, we go to t_1; if $x = \alpha$ and $f(\alpha) = 0$, we go to t_0; otherwise, we proceed to check the next string.

More precisely, let $\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(2^m)}$ be the lexicographic enumeration of $\{0,1\}^m$. For each $j \in \{1, \ldots, 2^m\}$, we create a checking gadget for $\alpha^{(j)}$ as follows:

The gadget consists of m layers. At layer i of gadget j, we query bit x_i:

- If $x_i = \alpha_i^{(j)}$, proceed to layer $i+1$ of the same gadget
- If $x_i \neq \alpha_i^{(j)}$, jump to layer 1 of gadget $j+1$ (or to t_0 if $j = 2^m$)

After completing all m layers of gadget j (meaning $x = \alpha^{(j)}$), we transition to $t_{f(\alpha^{(j)})}$.

This construction uses width 3 (nodes s, t_0, t_1, plus one node per layer in the current gadget, but we can reuse the same three nodes by appropriate labeling). The total length is $m \cdot 2^m$ since we have 2^m gadgets, each of length m, giving length $O(m2^m)$. □

N.7.4 Gemini

Proof. To prove the existence of a branching program for $f : \{0,1\}^m \to \{0,1\}$ with length m and width 2^m, we construct a leveled branching program based on a complete decision tree. Each level $i \in \{0, \ldots, m\}$ corresponds to the i-th variable x_i of the input. At level i, there are 2^i nodes representing all possible assignments to the first i variables $(x_1, \ldots, x_i)$. Since $i \leq m$, the maximum number of nodes at any level is 2^m. From a node at level i representing a prefix $u \in \{0,1\}^i$, the 0-edge leads to a node at level $i+1$ representing $u0$, and the 1-edge leads to a node at level $i+1$ representing $u1$. At the final level m, a node representing $w \in \{0,1\}^m$ is labeled with the output $f(w) \in \{0,1\}$. The width is $\max_i 2^i = 2^m$ and the length is m.

To prove the existence of a branching program for f with width 3 and length $O(m2^m)$, we utilize the construction of a program that computes f via its Disjunctive Normal Form (DNF). Any function $f : \{0,1\}^m \to \{0,1\}$ can be written as

$$f(x_1, \ldots, x_m) = \bigvee_{w \in f^{-1}(1)} \left(\bigwedge_{j=1}^{m} \ell_j \right)$$

where $\ell_j = x_j$ if $w_j = 1$ and $\ell_j = \neg x_j$ if $w_j = 0$. Each minterm (conjunction of m literals) corresponds to exactly one input w where $f(w) = 1$. There are at most 2^m such minterms.

We construct the branching program as a sequence of blocks, where each block checks one minterm. A single minterm $\bigwedge_{j=1}^{m} \ell_j$ can be computed by a width-2 branching program of length m: starting at an initial node s, if a literal ℓ_j is false, the edge points to a "reject" sink; if it is true, it moves to the next node in the chain. To compute the disjunction of these minterms using width 3, we designate one node as the "current result" state. Let the three nodes at each level be $\{A, B, C\}$. We use node A as a path to the final 1-accepting state if any previous minterm has evaluated to true. If the current minterm is being evaluated and all previous were false, the computation stays within nodes B and C.

Specifically, for each minterm M_i for $i = 1, \ldots, |f^{-1}(1)|$, we construct a sub-program: 1. If node A is reached (meaning a previous M_k was 1), stay in A regardless of subsequent variables. 2. If in B, evaluate M_i. If $M_i(x) = 1$, move to A. If $M_i(x) = 0$, move to B at the start of the evaluation for M_{i+1}. 3. Node C acts as a temporary sink for failed literal checks within the current minterm evaluation before resetting for the next minterm. Each minterm block uses m length. Since there are at most 2^m minterms, the total length is $O(m2^m)$ and the width is 3. □

N.7.5 Grok

Proof. A branching program for a function $f : \{0,1\}^m \to \{0,1\}$ is a leveled directed acyclic graph with nodes partitioned into layers $0, 1, \ldots, L$, where layer 0 contains a single source node, non-sink nodes in layers 1 to $L-1$ each query some input variable x_i ($1 \leq i \leq m$) and have two outgoing edges labeled 0 and 1 to nodes in the next layer, and nodes in layer L are sinks labeled 0 or 1. The length is L, and the width is the maximum number of nodes in any layer. On input $x \in \{0,1\}^m$, the computation follows the unique path from the source consistent with the bits of x, and the output is the label of the reached sink in layer L.

First, we construct a branching program of length m and width 2^m. Define layers 0 to m. Layer 0 has one node v_ϵ. For $k = 0$ to $m-1$, layer k has 2^k nodes, each labeled by a distinct string $\sigma \in \{0,1\}^k$. The node v_σ in layer k queries x_{k+1}, with the b-edge ($b \in \{0,1\}$) going to $v_{\sigma b}$ in layer $k+1$. Nodes in layer m (labeled by full assignments $\alpha \in \{0,1\}^m$) are sinks labeled $f(\alpha)$. The maximum number of nodes in a layer is 2^m (in layer m), so the width is 2^m. The length is m. On input x, the path reaches v_x in layer m and outputs $f(x)$.

Next, we construct a branching program of width 3 and length $O(m2^m)$. Let $S = \{a \in \{0,1\}^m \mid f(a) = 1\}$, so $|S| \leq 2^m$. Enumerate $S = \{a^1, \ldots, a^k\}$ with $k = |S|$. We build k blocks, each of $m+1$ layers (m check layers and 1 decision layer), for total length $(m+1)k + 1 = O(m2^m)$ (adding one initial layer if needed).

Each layer has three nodes: p (potential match for current assignment), s (skipped current assignment), and acc (accepted). The node acc is accepting in the final layer; reaching any other node in the final layer

rejects. In every layer, the node acc queries x_1 (arbitrarily) and both edges go to acc in the next layer.

The source in layer 0 goes to p in the first layer of block 1 (on both labels, querying arbitrarily).

For each block $j = 1$ to k corresponding to a^j:

The block consists of m check layers followed by one decision layer.

For the ℓ-th check layer in the block ($\ell = 1$ to m, checking bit position ℓ):

- Node p queries x_ℓ.

The edge labeled a^j_ℓ goes to p in the next layer (match).

The other edge goes to s in the next layer (mismatch).

- Node s queries x_1, both edges to s in the next layer.

In the decision layer of the block:

- Node p (matched all m bits, so input equals a^j and $f(a^j) = 1$) queries x_1, both edges to acc in the next layer.

- Node s queries x_1, both edges to p in the first check layer of block $j + 1$ (or to a rejecting node in the final layer if $j = k$).

After the last block's decision layer, paths reaching acc in the final layer accept, others reject.

The width is 3. On any input x with $f(x) = 0$, no block fully matches, so the path never enters acc persistently and rejects. If $f(x) = 1$, then $x = a^j$ for some j; in prior blocks the path enters s at the first mismatch (or starts fresh), and in block j it stays in p through all checks and transitions to acc in the decision, staying in acc thereafter and accepting.

Thus, the construction is correct. □

N.8 Problem 8

N.8.1 Problem Statement

Theorem N.8 (Exercise 12.17)**.** *Consider the element distinctness function,* ED*, of Example 7.14. Prove that any oblivious branching program for* ED *requires either* $\Omega(m \log m)$ *length or exponential width.*

N.8.2 GPT

Proof. We are tasked with proving that any oblivious branching program for the element distinctness function, ED, requires either $\Omega(m \log m)$ length or exponential width.

Let ED be the function defined in Example 7.14, where the input is a list of m elements, and the output is 1 if all elements are distinct, and 0 if there are any repetitions. We will consider an oblivious branching program B that computes ED, and we aim to show that the length of B must be at least $\Omega(m \log m)$ or its width must be exponential in m.

Step 1: Analysis of the structure of an oblivious branching program An oblivious branching program is a special kind of branching program where the sequence of tests is fixed and does not depend on the previous inputs. That is, each layer of the program performs the same test on each input bit, and the tests are performed in the same order for every input. Specifically, for ED, the branching program operates on the bits of the input array $x = (x_1, x_2, \ldots, x_m)$, where x_i is the i-th element.

Step 2: Complexity of computing ED The function ED requires the comparison of all pairs of elements to detect duplicates. This can be thought of as a problem of checking whether there exists a pair $i \neq j$ such that $x_i = x_j$. A straightforward method would involve comparing every pair of elements, resulting in $\binom{m}{2} = \frac{m(m-1)}{2}$ comparisons. Hence, the size of the input space is 2^m for a string of length m, and any oblivious branching program for ED must somehow distinguish between these 2^m possibilities.

Step 3: Lower bounds on length and width To prove the lower bounds on the length or width of an oblivious branching program, we apply known results from the theory of branching programs.

1. **Lower bound on length**: Any oblivious branching program for ED must be able to distinguish between all input strings, and the number of distinct input strings is 2^m. A branching program has at most w possible configurations of the vertices in each layer, where w is the width of the program. Since the program must distinguish between 2^m possible inputs, there must be at least $\log_w(2^m) = m \log_w 2$ layers, where w is the width of the program. Hence, if the width is fixed, the length of the program must be at least $\Omega(m \log m)$.

2. **Lower bound on width**: If the length of the program is fixed, the program must use a sufficiently large width to distinguish between all possible inputs. If the program has L layers, then the number of distinct paths through the program is bounded by w^L. Since there are 2^m possible inputs, we must have $w^L \geq 2^m$, which implies that $w \geq 2^{m/L}$. Thus, if the length of the program is L, the width must be exponential in m.

Step 4: Conclusion From the above two cases, we conclude that for any oblivious branching program computing ED, either the length must be at least $\Omega(m \log m)$, or the width must be exponential in m. This completes the proof. □

N.8.3 Claude

Proof. We prove that any oblivious branching program computing the element distinctness function $\text{ED} : [m]^n \to \{0, 1\}$ (where $\text{ED}(x_1, \ldots, x_n) = 1$ if all elements are distinct and 0 otherwise) must have either length $\Omega(m \log m)$ or exponential width.

Let B be an oblivious branching program of length L and width W computing ED. We proceed by contradiction. Assume that $L = o(m \log m)$ and $W = 2^{o(m)}$.

For the element distinctness problem with $n = m$ elements from domain $[m]$, consider the communication complexity lower bound approach. We use a fooling set argument combined with the information-theoretic method.

Define a partition of the input variables into two sets: $X = \{x_1, \ldots, x_{m/2}\}$ and $Y = \{x_{m/2+1}, \ldots, x_m\}$. Consider the two-party communication problem where Alice receives X and Bob receives Y, and they must determine if all m elements are distinct.

By the oblivious property, the branching program reads variables in a fixed order. Let L_X denote the number of layers reading variables from X and L_Y denote the number of layers reading variables from Y, so $L = L_X + L_Y$.

At any intermediate layer i between reading X and Y, the state of the computation is determined by a node in the branching program. The number of possible states is at most W.

Consider the fooling set $\mathcal{F}$ consisting of all permutations $\pi : [m] \to [m]$ restricted to inputs where the first $m/2$ positions contain distinct values and the last $m/2$ positions contain distinct values. For any two distinct permutations $\pi_1, \pi_2 \in \mathcal{F}$, if they agree on X but differ on Y (or vice versa), then at least one of the combined inputs must have a repeated element.

The information flow through the branching program can be bounded as follows. After processing X, the branching program is in one of at most W states. Each state must encode sufficient information to distinguish between the $(m/2)!$ possible distinct assignments to X.

By the pigeonhole principle, if $W < (m/2)!$, then there exist two distinct assignments $x, x' \in [m]^{m/2}$ (both with all distinct elements) that lead to the same state s after processing X.

Now consider assignments to Y. For the branching program to correctly compute ED, it must be able to determine, for each assignment $y \in [m]^{m/2}$ with distinct elements, whether y shares any element with x or x'. However, since $x \neq x'$, there exists an element $a \in x$ such that $a \notin x'$ (or vice versa).

The number of assignments y that avoid all elements in x is $(m/2)! \cdot \binom{m/2}{m/2}$, while the number avoiding all elements in x' is different. The branching program starting from state s must distinguish all these cases using only L_Y layers and width W.

By an information-theoretic counting argument, the total number of distinct computation paths from state s is at most W^{L_Y}. This must be at least $(m/2)!$ to handle all possible distinct assignments to Y that interact differently with x versus x'.

Therefore, we require $W^{L_Y} \geq (m/2)!$, which gives $L_Y \geq \frac{\log((m/2)!)}{\log W} = \frac{\Omega(m \log m)}{\log W}$.

Similarly, $L_X \geq \frac{\Omega(m \log m)}{\log W}$.

Thus, $L = L_X + L_Y \geq \frac{\Omega(m \log m)}{\log W}$.

If $W = 2^{o(m)}$, then $\log W = o(m)$, which implies $L \geq \frac{\Omega(m \log m)}{o(m)} = \Omega(\log m) \cdot \omega(1) = \omega(\log m)$. More precisely, $L = \Omega(m \log m)$ when $\log W = O(m)$.

Conversely, if $L = o(m \log m)$, then $\log W \geq \frac{\Omega(m \log m)}{o(m \log m)} = \omega(1) \cdot m = \omega(m)$, giving $W = 2^{\omega(m)}$, which is exponential.

Therefore, any oblivious branching program for ED requires either length $\Omega(m \log m)$ or exponential width. □

N.8.4 Gemini

Proof. Let $X = (x_1, \ldots, x_m)$ be the input to the element distinctness function ED_m, where each $x_i \in \{0,1\}^k$ for $k = 2 \log_2 m$. The function $\mathrm{ED}_m(X) = 1$ if and only if for all $i \neq j$, $x_i \neq x_j$. Let B be an oblivious branching program of length L and width W computing ED_m.

Assume for contradiction that $L < \epsilon m \log m$ for a sufficiently small constant $\epsilon > 0$ and $W < 2^{\delta m}$ for a sufficiently small constant $\delta > 0$. An oblivious branching program of length L defines a sequence of variables $(z_1, z_2, \ldots, z_L)$ where each $z_t \in \{x_1, \ldots, x_m\}$. Let $I_j = \{t \in \{1, \ldots, L\} : z_t = x_j\}$ be the set of time steps at which variable x_j is queried. Since $\sum_{j=1}^{m} |I_j| = L < \epsilon m \log m$, by an averaging argument, at least $m/2$ variables x_j are queried at most $2\epsilon \log m$ times. Let $J \subseteq \{1, \ldots, m\}$ be the set of indices of these variables, where $|J| \geq m/2$.

We utilize a communication complexity argument by partitioning J into two sets A and B such that $|A| = |B| = m/4$. We consider the information flow between A and B. In an oblivious branching program, the state at any time t must capture all necessary information about the variables queried before t to correctly compute the function later.

Let $t_1 < t_2 < \cdots < t_L$ be the levels of the program. For any cut at time t, the number of possible states is at most W. This implies that the amount of information that can pass through a level is at most $\log_2 W$ bits. If we consider the variables in A as being held by Player 1 and variables in B by Player 2, the branching program induces a communication protocol.

The total number of bits communicated in the protocol is at most $K \cdot \log_2 W$, where K is the number of times the variable sequence (z_t) switches between a variable in A and a variable in B. However, for an oblivious program, the variables are fixed in sequence. The total number of "intervals" of consecutive queries to variables within the same set (A or B) is bounded by the number of queries.

For the element distinctness function, it is a known result from communication complexity that to determine if $x_A \cap x_B = \emptyset$ (where $x_A = \{x_i : i \in A\}$ and $x_B = \{x_j : j \in B\}$), the players must exchange $\Omega(m \log m)$ bits of information. In our construction, each variable $x_j \in A \cup B$ is queried at most $2\epsilon \log m$ times. The total number of crossings between A and B in the sequence (z_t) is at most L.

The information transmitted across all boundaries must be at least the communication complexity of ED. Thus, we have the inequality:

$$L \cdot \log_2 W \geq \Omega(m \log m)$$

If $L = O(m)$, then $\log_2 W \geq \Omega(\log m)$, which allows for polynomial width. However, if we restrict the number of times each variable is queried (the "passes" over the data), the constraint becomes tighter. Specifically, for element distinctness, the property that $L \cdot \log_2 W = \Omega(m^2)$ for certain restricted models applies, but in the general oblivious branching program case, the standard bottleneck identity implies that if L is not significantly larger than $m \log m$, specifically if $L = o(m \log m)$, the capacity of the levels must grow exponentially to maintain the distinctness information.

Therefore, if $L < \epsilon m \log m$, the number of states W must satisfy $\log_2 W \geq \delta m$ for some $\delta > 0$, implying $W \geq 2^{\delta m}$. This concludes that any oblivious branching program for ED requires either $\Omega(m \log m)$ length or exponential width. □

N.8.5 Grok

O Problems in Chapter 13

O.1 Problem 1

O.1.1 Problem Statement

Theorem O.1 (13.8)**.** *Obtain a similar deterministic amplification for algorithms with two-sided error.*

O.1.2 GPT

Proof. Let A be a randomized algorithm with two-sided error. This means that for any input x, the algorithm $A(x)$ has the following error probabilities:

- With probability at least $1 - \epsilon$, $A(x)$ outputs the correct answer. - With probability at most ϵ, $A(x)$ outputs the wrong answer.

We wish to amplify the probability of success for A by repeating the algorithm multiple times and taking a majority vote of the outcomes. Specifically, we run A independently k times on the same input x, and output the majority of the k results. The goal is to show that this majority-vote strategy decreases the error probability of the resulting amplified algorithm.

Let $A_1, A_2, \ldots, A_k$ be k independent executions of the algorithm A, and let X_i denote the outcome of the i-th execution, where $X_i = 1$ if the i-th execution is correct and $X_i = 0$ if it is incorrect. Each X_i is a Bernoulli random variable with success probability $1 - \epsilon$. The majority vote algorithm outputs 1 if more than half of the X_i are 1, and outputs 0 otherwise. We want to show that the probability of error for the majority vote algorithm is small.

Define $S = \sum_{i=1}^{k} X_i$, the total number of correct outputs among the k executions. The majority vote algorithm outputs 1 if $S > \frac{k}{2}$, and outputs 0 otherwise. The error occurs if the majority vote is incorrect, i.e., if $S \leq \frac{k}{2}$ when the correct answer is 1, or $S > \frac{k}{2}$ when the correct answer is 0.

Let p be the probability that the majority vote is incorrect. This probability is equivalent to the tail probability of the binomial distribution:

$$p = \mathbb{P}\left(S \leq \frac{k}{2} \mid \text{correct answer is } 1\right) + \mathbb{P}\left(S > \frac{k}{2} \mid \text{correct answer is } 0\right)$$

Since each X_i is independent and follows a Bernoulli distribution with success probability $1 - \epsilon$, the sum S follows a binomial distribution: $S \sim \text{Binomial}(k, 1 - \epsilon)$. Using Chernoff's bound, we can bound the probability of error as follows:

$$\mathbb{P}(S \leq \frac{k}{2}) \leq \exp\left(-\frac{k\epsilon^2}{2}\right)$$

Thus, the probability of error for the majority vote algorithm is at most:

$$p \leq 2\exp\left(-\frac{k\epsilon^2}{2}\right)$$

By choosing $k = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, we can ensure that the probability of error is at most δ, i.e.,

$$p \leq \delta$$

Therefore, by repeating k independent executions of the algorithm A and taking the majority vote, we obtain a new algorithm with error probability at most δ, which is the desired deterministic amplification for algorithms with two-sided error. □

O.1.3 Claude

Proof. Let A be a randomized algorithm with two-sided error that decides a language L with error probability at most $\epsilon < \frac{1}{2}$. That is, for any input x:

$$\Pr[A(x) \text{ is correct}] \geq 1 - \epsilon > \frac{1}{2}.$$

We construct a deterministic amplification procedure as follows. For a fixed input x of length n, run algorithm A on x using k independent random strings $r_1, r_2, \ldots, r_k$, where $k = 2t + 1$ is odd. Let $A(x; r_i)$ denote the output of A on input x using random string r_i. Define the amplified algorithm A' by:

$$A'(x) = \text{majority}\{A(x; r_1), A(x; r_2), \ldots, A(x; r_k)\}.$$

For a fixed input x, let X_i be the indicator random variable where $X_i = 1$ if $A(x; r_i)$ is correct and $X_i = 0$ otherwise. Then $\Pr[X_i = 1] \geq 1 - \epsilon$ and the X_i are independent. Let $X = \sum_{i=1}^{k} X_i$ be the number of correct answers.

The algorithm $A'(x)$ errs if and only if the majority of the k runs produce incorrect answers, which occurs when $X \leq t = \frac{k-1}{2}$. By the Chernoff bound, for $\mu = \mathbb{E}[X] = k(1-\epsilon)$ and $\delta = \frac{\epsilon - \frac{1}{2}}{1-\epsilon}$, we have $(1-\delta)\mu = k \cdot \frac{1}{2}$. Since $1 - \epsilon > \frac{1}{2}$, we have $\delta \in (0, 1)$.

Applying the Chernoff bound:

$$\Pr[A'(x) \text{ errs}] = \Pr[X \leq t] = \Pr\left[X \leq \frac{k}{2}\right] \leq \exp\left(-\frac{\delta^2 \mu}{2}\right) = \exp\left(-\frac{k(1-\epsilon)(\epsilon - \frac{1}{2})^2}{2(1-\epsilon)^2}\right).$$

Simplifying:

$$\Pr[A'(x) \text{ errs}] \leq \exp\left(-\frac{k(\epsilon - \frac{1}{2})^2}{2(1-\epsilon)}\right).$$

For any desired error probability $\delta' > 0$, choose:

$$k = \left\lceil \frac{2(1-\epsilon)\ln(1/\delta')}{(\epsilon - \frac{1}{2})^2} \right\rceil.$$

Then $\Pr[A'(x) \text{ errs}] \leq \delta'$ for all inputs x.

To derandomize, enumerate all possible k-tuples of random strings $(r_1, \ldots, r_k)$ from the random string space of A. For each tuple, compute the majority vote. Since the error probability can be made arbitrarily small by choosing k large enough, there exists at least one k-tuple that produces the correct answer for all inputs. This k-tuple can be found by exhaustive search, yielding a deterministic algorithm with zero error. □

O.1.4 Gemini

Proof. Let L be a language in $\mathbf{BPP}$. By definition, there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ and a polynomial $r(n)$ such that for every input $x \in \{0,1\}^n$, $\mathcal{A}$ uses $r(n)$ random bits and satisfies

$$\Pr_{\rho \in \{0,1\}^{r(n)}}[\mathcal{A}(x, \rho) \neq \chi_L(x)] \leq \frac{1}{3},$$

where $\chi_L(x)$ is the characteristic function of L. We seek an algorithm $\mathcal{A}'$ that achieves an error probability of at most 2^{-k} using $r(n) + O(k)$ random bits.

Let $G = (V, E)$ be a strongly explicit d-regular expander graph family with $V = \{0,1\}^{r(n)}$ and normalized second eigenvalue $\lambda < 1$. Such graphs exist with constant degree d (e.g., $d = 16$).

We define the amplified algorithm $\mathcal{A}'(x)$ as follows: 1. Select a starting vertex v_1 uniformly at random from V. 2. Perform a random walk of length $t - 1$ on G starting at v_1 to generate a sequence of vertices $v_1, v_2, \ldots, v_t$. 3. Compute $y_i = \mathcal{A}(x, v_i)$ for each $i \in \{1, \ldots, t\}$. 4. Output the majority value of the sequence $(y_1, \ldots, y_t)$.

To analyze the error probability, fix an input x. Let $B_x \subset V$ be the set of "bad" random strings, defined as $B_x = \{\rho \in V \mid \mathcal{A}(x, \rho) \neq \chi_L(x)\}$. By the definition of $\mathcal{A}$, the density of this set is $\mu(B_x) = \frac{|B_x|}{|V|} \leq \frac{1}{3}$.

Let $f : V \to \{0,1\}$ be the indicator function of the set B_x, such that $f(v) = 1$ if $v \in B_x$ and $f(v) = 0$ otherwise. The algorithm $\mathcal{A}'$ outputs the incorrect value if and only if the majority of the random strings $v_1, \ldots, v_t$ lie in B_x. That is, $\mathcal{A}'$ errs if

$$\sum_{i=1}^{t} f(v_i) \geq \frac{t}{2}.$$

Dividing by t, this condition is equivalent to

$$\frac{1}{t}\sum_{i=1}^{t} f(v_i) \geq \frac{1}{2}.$$

The expectation of the average value of f over the random walk is given by the density of B_x:

$$\mathbb{E}\left[\frac{1}{t}\sum_{i=1}^{t} f(v_i)\right] = \mu(B_x) \leq \frac{1}{3}.$$

Consequently, for the algorithm to err, the empirical average must deviate from the expectation by at least $\frac{1}{2} - \frac{1}{3} = \frac{1}{6}$. We invoke the Expander Chernoff Bound (Gillman's Bound), which states that for a random walk $v_1, \ldots, v_t$ on a graph with spectral expansion λ, and any function $f : V \to [0,1]$ with mean μ,

$$\Pr\left[\left|\frac{1}{t}\sum_{i=1}^{t} f(v_i) - \mu\right| \geq \epsilon\right] \leq 2\exp\left(-\frac{(1-\lambda)\epsilon^2 t}{4}\right).$$

Setting $\epsilon = \frac{1}{6}$, we obtain

$$\Pr[\text{Error}] = \Pr\left[\frac{1}{t}\sum_{i=1}^{t} f(v_i) \geq \frac{1}{2}\right] \leq \Pr\left[\left|\frac{1}{t}\sum_{i=1}^{t} f(v_i) - \mu(B_x)\right| \geq \frac{1}{6}\right].$$

Substituting into the bound:

$$\Pr[\text{Error}] \leq 2\exp\left(-\frac{(1-\lambda)}{144}t\right).$$

Since λ is a constant strictly less than 1 (dependent only on the explicit expander construction), there exists a constant $c > 0$ such that the error probability is bounded by 2^{-ct}. To achieve a target error probability of 2^{-k}, we choose $t = \Theta(k)$.

The number of random bits required by $\mathcal{A}'$ is the sum of the bits needed to choose the start vertex v_1 and the bits needed to determine the $t-1$ steps of the walk. Since $v_1 \in \{0,1\}^{r(n)}$ and the degree of G is d, each step requires $\log d$ bits. The total randomness is

$$r(n) + (t-1)\log d = r(n) + O(k).$$

Thus, we have obtained a deterministic amplification for the two-sided error algorithm $\mathcal{A}$ that reduces the error to 2^{-k} using only $O(k)$ additional random bits. □

O.1.5 Grok

Proof. We formalize the theorem as follows.

There exists a universal constant $C > 0$ such that if there is a polynomial-time probabilistic algorithm A that on input $x \in \{0,1\}^n$ uses $r(n)$ random bits and satisfies

$$\Pr[A(x) = f(x)] \geq \frac{2}{3}$$

for a function f, then for every $t(n) \geq 1$ there exists a polynomial-time probabilistic algorithm A' using at most $r(n) + Ct(n)$ random bits such that

$$\Pr[A'(x) = f(x)] \geq 1 - 2^{-t(n)}.$$

The proof uses explicit constant-degree spectral expanders and the expander Chernoff bound.

There exist constants $d \geq 3$ and $\lambda < 1$ (with $\gamma = 1 - \lambda > 0$) and an explicit family of d-regular graphs $\{G_m = (V_m, E_m)\}_{m \geq 1}$ with $|V_m| = 2^m$ such that the normalized adjacency matrix of G_m has largest eigenvalue 1 and all other eigenvalues satisfy $|\mu| \leq \lambda$ for every m. The neighbor function is computable in polynomial time.

Fix $m = r = r(n)$ and let $G = G_r$ with vertex set $V = \{0,1\}^r$ (identified naturally with $[2^r]$).

The amplified algorithm A' on input x and parameter $t = t(n)$ operates as follows. Let $k = C_1 t$ where $C_1 > 0$ is a sufficiently large constant to be determined (depending only on d and γ).

- Sample a uniform random $s_0 \in V$ (using r bits). - For $i = 1, \dots, k-1$, sample a uniform random neighbor index $j_i \in [d]$ (using $\lceil \log_2 d \rceil$ bits each) and set s_i to be the j_i-th neighbor of s_{i-1}. - Compute $b_i = A(x, s_i)$ for $i = 0, \dots, k-1$. - Output the majority value of $\{b_0, \dots, b_{k-1}\}$.

The total randomness is $r + (k-1)\lceil \log_2 d \rceil \le r + Ct$ for a universal C (since d is constant and C_1 universal).

The running time is polynomial because $k = O(t)$ is polynomial (as $t(n) \ge 1$ is arbitrary but the original algorithm is polynomial-time), each A call is polynomial-time, and neighbors are computable in polynomial time.

For correctness, fix x and define $g : V \to \{0, 1\}$ by $g(s) = A(x, s)$. Let

$$\mu = \mathop{\mathbb{E}}_{s \text{uniform} V}[g(s)] = \Pr_{s \text{uniform} V}[A(x, s) = 1].$$

If $f(x) = 1$ then $\mu \ge 2/3$, and if $f(x) = 0$ then $\mu \le 1/3$.

Let $X_i = g(s_i)$ and $\bar{X} = k^{-1} \sum_{i=0}^{k-1} X_i$. The algorithm outputs 1 if $\bar{X} \ge 1/2$ and 0 otherwise (ties broken arbitrarily; their probability is negligible for the chosen parameters).

Since G is regular and s_0 is uniform, the stationary distribution is uniform and thus the marginal distribution of each s_i is uniform over V. Hence $\mathbb{E}[X_i] = \mu$ for all i and $\mathbb{E}[\bar{X}] = \mu$.

We apply the following concentration bound for random walks on spectral expanders.

There exists a constant $c = c(d, \gamma) > 0$ such that for any function $h : V \to [0, 1]$ and any walk length $k \ge 1$,

$$\Pr\left[\left|k^{-1} \sum_{i=0}^{k-1} h(s_i) - \mu_h\right| \ge \delta\right] \le 2 \exp(-c\delta^2 k),$$

where $\mu_h = \mathbb{E}_{v \text{uniform} V}[h(v)]$.

(This is a standard expander Chernoff bound; see e.g. Gillman (1993) or later improvements.)

Applying this to $h = g$ with $\delta = 1/6$,

$$\Pr[|\bar{X} - \mu| \ge 1/6] \le 2 \exp(-c'k),$$

where $c' = c/36 > 0$.

Suppose $f(x) = 1$, so $\mu \ge 2/3$. If $|\bar{X} - \mu| < 1/6$ then $\bar{X} > 2/3 - 1/6 = 1/2$, so the majority is 1 and the output is correct. Thus the error probability is at most $2 \exp(-c'k)$.

Similarly, if $f(x) = 0$ then $\mu \le 1/3$ and if $|\bar{X} - \mu| < 1/6$ then $\bar{X} < 1/3 + 1/6 = 1/2$, so the majority is 0 and the output is correct. Again, error probability at most $2 \exp(-c'k)$.

Choosing C_1 sufficiently large (depending on c') ensures $c' C_1 t \ge t \ln 2 + \ln 4$ for all $t \ge 1$, so $2 \exp(-c'k) \le 2^{-t}$.

Thus the success probability is at least $1 - 2^{-t}$. □

O.2 Problem 2

O.2.1 Problem Statement

Theorem O.2. *The simulation using the inner product function $\mathbb{P}$ works for every fixed $\delta > 0$.*

O.2.2 GPT

Proof. Let $\delta > 0$ be fixed. We aim to show that the simulation using the inner product function $\mathbb{P}$ works for every fixed $\delta > 0$.

To begin, consider the definition of the inner product function $\mathbb{P}$. By the properties of the inner product, for any two vectors $\mathbf{v}$ and $\mathbf{w}$, we have the following basic properties:

$$\langle \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{w}, \mathbf{v} \rangle$$

and

$$\langle \mathbf{v}, \mathbf{v} \rangle \ge 0,$$

with equality if and only if $\mathbf{v} = 0$. These properties hold for any vectors in the relevant vector space, and we assume that the inner product function $\mathbb{P}$ satisfies these basic axioms.

Next, we establish that the simulation procedure based on $\mathbb{P}$ behaves correctly for every fixed $\delta > 0$. Let us assume that the simulation fails to work for some $\delta > 0$. This would imply the existence of a contradiction in the behavior of $\mathbb{P}$ under the simulation.

To reach a contradiction, suppose there exists an $\epsilon > 0$ such that for some vectors $\mathbf{v}$ and $\mathbf{w}$, the inequality

$$|\langle \mathbf{v}, \mathbf{w} \rangle - \mathbb{P}(\mathbf{v}, \mathbf{w})| \geq \epsilon$$

holds. However, by the definition of the simulation, and given that $\mathbb{P}$ is defined to behave as an inner product, we expect that for any $\mathbf{v}$ and $\mathbf{w}$, the inner product approximation provided by $\mathbb{P}$ should satisfy the condition

$$|\langle \mathbf{v}, \mathbf{w} \rangle - \mathbb{P}(\mathbf{v}, \mathbf{w})| < \delta.$$

Thus, by the assumption that the simulation fails for some δ, we would have that the error is greater than or equal to δ, which contradicts the assumption that $\mathbb{P}$ approximates the inner product correctly to within δ for all $\mathbf{v}$ and $\mathbf{w}$.

Therefore, the assumption that the simulation fails for some $\delta > 0$ must be false. Hence, the simulation using the inner product function $\mathbb{P}$ works for every fixed $\delta > 0$.

□

O.2.3 Claude

Proof. Let $\delta > 0$ be fixed and arbitrary. We must show that the simulation using the inner product function $\mathbb{P}$ satisfies the required properties for this choice of δ.

First, we establish that the inner product function $\mathbb{P} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ is well-defined and continuous on the Hilbert space $\mathcal{H}$. By the Cauchy-Schwarz inequality, for any $x, y \in \mathcal{H}$, we have $|\mathbb{P}(x, y)| = |\langle x, y \rangle| \leq \|x\| \cdot \|y\|$, which ensures boundedness.

Next, we verify the simulation property. For any state ρ in the system and any observable A, the simulated expectation value using $\mathbb{P}$ is given by

$$\mathbb{E}_{\mathbb{P}}[A] = \text{Tr}(\rho A) + \varepsilon(\delta),$$

where $\varepsilon(\delta)$ represents the error term dependent on δ.

We now bound the error term. By construction of the simulation protocol, the error satisfies

$$|\varepsilon(\delta)| \leq C \cdot \delta \cdot \|A\|_{\text{op}},$$

where $C > 0$ is a universal constant independent of δ and $\|A\|_{\text{op}}$ denotes the operator norm of A.

Since $\delta > 0$ is fixed, we have $|\varepsilon(\delta)| < \infty$, which establishes that the simulation produces finite, well-defined results.

To verify completeness, we must show that for any measurement outcome in the original system, there exists a corresponding outcome in the simulated system with probability differing by at most $O(\delta)$. Let $\{P_i\}$ be a complete set of projection operators. For each i, the probability in the original system is $p_i = \text{Tr}(\rho P_i)$, while in the simulated system it is $\widetilde{p}_i = \text{Tr}(\rho P_i) + \varepsilon_i(\delta)$. By the previous bound and normalization, we have

$$\sum_i |\widetilde{p}_i - p_i| = \sum_i |\varepsilon_i(\delta)| \leq C'\delta,$$

for some constant $C' > 0$.

Finally, we verify that the simulation preserves the inner product structure up to error δ. For any $|\psi\rangle, |\phi\rangle \in \mathcal{H}$, the simulated inner product satisfies

$$|\mathbb{P}_{\text{sim}}(\psi, \phi) - \langle \psi | \phi \rangle| \leq \delta \cdot \|\psi\| \cdot \|\phi\|.$$

This follows from the triangle inequality and the construction of $\mathbb{P}_{\text{sim}}$ as an approximation to the true inner product with precision parameter δ.

Therefore, for every fixed $\delta > 0$, the simulation using the inner product function $\mathbb{P}$ is well-defined, produces bounded errors, and faithfully reproduces the quantum mechanical predictions within tolerance $O(\delta)$. □

O.2.4 Gemini

Proof. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a bipartite function and let $\text{IP}(x,y) = \sum_{i=1}^{n} x_i y_i \pmod 2$ denote the inner product function over $\mathbb{F}_2^n$. We consider the simulation of a communication protocol for a function $g = f \circ \text{IP}$ and aim to show that for every fixed $\delta > 0$, the inner product function is a valid gadget for lifting theorems from query complexity to communication complexity.

Let $U, V \subseteq \{0,1\}^n$ be the sets of inputs held by Alice and Bob respectively. The discrepancy of the inner product function, denoted $\text{disc}(\text{IP})$, is known to be bounded by $2^{-n/2}$. Specifically, for any rectangles $R = A \times B$, the bias of IP on R satisfies:

$$|\mathbb{P}_{(x,y)\in R}[\text{IP}(x,y) = 0] - \mathbb{P}_{(x,y)\in R}[\text{IP}(x,y) = 1]| \leq \frac{\sqrt{|A||B|}}{2^n} \leq 2^{-n/2}$$

This small discrepancy property is the fundamental requirement for the simulation theorem.

To show the simulation works for a fixed $\delta > 0$, we consider the rank-sum or corruption lemma approach. Let D be a distribution on the inputs of f such that any decision tree for f of depth d has success probability at most $1 - \epsilon$. We define the lifted function $F = f \circ \text{IP}^k$. The simulation theorem states that the communication complexity $D(F)$ is related to the query complexity $D^{\text{dt}}(f)$ by:

$$D(F) \geq \Omega(D^{\text{dt}}(f) \cdot \log n)$$

provided the gadget IP has sufficiently small discrepancy.

For the inner product function $\text{IP} : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$, the number of bits communicated per query simulated is $O(k)$. For the simulation to be robust against an adversary's strategy, we require that for every rectangle R in the communication protocol that is "large" (i.e., $|R| \geq 2^{2n(1-\delta)}$), the distribution of the outputs of the gadget IP is nearly uniform.

By the Vazirani-Levin theorem or the Lindström-Gessel-Viennot lemma applied to the parity of bits, the bias of any non-trivial linear combination of the inner product outputs $\bigoplus_{i\in S} \text{IP}(x^{(i)}, y^{(i)})$ for a set of coordinates S is bounded by $2^{-|S|k/2}$. For a fixed $\delta > 0$, we choose the gadget size $k = C \log n$ for a sufficiently large constant C.

The probability that the inner product gadget fails to simulate a query bit correctly within a rectangle of density $2^{-\delta n}$ is bounded by:

$$\epsilon_{\text{fail}} \leq 2^{\delta n} \cdot \text{disc}(\text{IP}) \leq 2^{\delta n - k/2}$$

By choosing $k \geq 2\delta n + O(1)$, the error probability becomes negligible. Since this holds for every fixed $\delta > 0$ by adjusting the constant factors in the gadget size relative to the input size, the simulation of the decision tree by the communication protocol is sound.

Furthermore, the consistency of the simulation relies on the fact that the inner product function satisfies the "thickness" property. For every fixed $\delta > 0$, a rectangle R in the communication space $(\{0,1\}^k \times \{0,1\}^k)^n$ that has measure at least $2^{-\delta nk}$ is "thick" in the sense that it contains a structured subset where the gadget outputs are essentially independent. Thus, the inner product function IP successfully facilitates the lifting of lower bounds for any $\delta > 0$. □

O.2.5 Grok

Proof. Let $\delta > 0$ be fixed. By the construction of the simulation, the inner product function $\mathbb{P}$ is defined such that all parameters depending on the confidence or accuracy level are functions of δ alone (e.g., number of samples, repetition counts, or precision requirements of the form $O(\text{poly}(1/\delta))$ or $O(\log(1/\delta))$).

Since $\delta > 0$ is fixed and independent of any other problem parameters (such as input size or dimension), these quantities are constant with respect to the underlying problem instance. The simulation therefore proceeds in finite (constant) time relative to the variable parameters and satisfies the required correctness

guarantees with probability at least $1 - \delta$ (or within additive/multiplicative error bounded in terms of δ, as appropriate to the specific setting).

As this holds for any arbitrarily chosen fixed $\delta > 0$, the simulation using the inner product function $\mathbb{P}$ works for every fixed $\delta > 0$. $\square$