Beyond Short Answers: Evaluating Long-Form Reasoning in LLMs via Communication Complexity

Anonymous ACL submission

Abstract

With the continuous development of large language models (LLMs), their capabilities in solving mathematical problems have attracted increasing attention from the research community. Bubeck et al. (Bubeck et al., 2025) present that GPT-5 can accelerate scientific progress and propose new ideas in interdisciplinary research. Additionally, Georgiev, Gómez-Serrano, Tao, and Wagner (Georgiev et al., 2025) utilize large language models as powerful tools for scalable mathematical exploration, enabling conjecture generation and pattern discovery that complement human mathematical reasoning. While recent advances have highlighted the strong potential of large language models in mathematical research, their capabilities on TCS problems remain underexplored, motivating the development of a benchmark to evaluate the understanding and reasoning abilities of large language models on TCS mathematical problems. In our work, we proposed a comprehensive benchmark for four state-of-the-art models: GPT-5-Thinking, Grok-4, Gemini-3-Pro-Thinking, and Claude-Sonnet-4.5, based on the Communication Complexity of Rao and Yehudayoff (Rao and Yehudayoff, 2020).

We have each model generate LaTeX-formatted proofs for the exercises and theorems in the book, and assess the results through a combination of LLM-based evaluation and use human annotators for verification. The results indicate that while leading models (Gemini) achieve accuracies of approximately 92.0%, other models attain only around 22.7%. Moreover, although cutting-edge models have reached a proficiency level suitable for graduate-level teaching assistance and mathematical formalization, substantial disparities persist in their reliability for rigorous mathematical derivations.

1 Introduction

With the continuous emergence of efficient mathematical computing tools, the field of mathematics has undergone a profound transformation (Wagner, 2021; Fawzi et al., 2022; Romera-Paredes et al., 2024). The role of these systems is no longer limited to improving the efficiency of symbolic computation or numerical experiments. The current large language models are beginning to be able to explore in large-scale combinatorial structures and geometric configuration spaces, automatically identifying previously unrecorded extreme constructions, and in some cases assisting in the generation and proof of new mathematical conjectures. These developments indicate that large language models may play an increasingly important role as computational assistance tools in modern mathematical research.

Meanwhile, large language models have made significant progress in formal reasoning and coding works (Fried et al., 2023; Nijkamp et al., 2023), such as program debugging (Liventsev et al., 2023), algorithm problem solving (Zelikman et al., 2023), and code performance optimization (Shypula et al., 2024). These abilities provide a crucial foundation for their involvement in mathematical research. However, when addressing open mathematical research problems, generating complete and verifiable new proofs or conjectures still requires novel ideas beyond existing knowledge, which remains challenging for current large language models. On the one hand, models may produce unreliable or poorly substantiated inferences; On the other hand, models' generation process is often constrained by training data, making it difficult to systematically and efficiently explore novel and complex mathematical theories. Recent research (Wang et al., 2025) has explored the integration of large language models with tree-based search methods, allowing models to more efficiently solve mathematical reasoning tasks.

Despite the growing prominence of large language models in mathematics-related tasks, there is still a relative lack of systematic, rigorous, and comprehensive evaluation of their advanced mathe-

matical abilities. It remains unclear whether these models genuinely grasp the core concepts of modern mathematics, particularly their ability to handle lengthy and complex proofs, maintain overall structural and logical coherence, and apply key theoretical ideas with clarity and precision. This raises a fundamental question:

Can large language models perform the long-form mathematical reasoning required in modern theoretical research?

To address this question, we propose a systematic evaluation framework based on the Communication Complexity (Rao and Yehudayoff, 2020) of classic TCS textbooks to assess the reasoning quality and formal rigor of the model on advanced theoretical computer science problems.

Communication Complexity covers core topics such as lower bound proofs under communication models, protocol design, information theory methods, and combinatorial techniques. The exercises at the end of each chapter require multi-step derivations within a strict computational step, and the construction of a complete formal proof. These exercises inherently possess characteristics such as deep reasoning, long argumentation span, and complex logical dependencies. Therefore, they provide a highly challenging and low-ambiguity testing environment for assessing whether large language models have mathematical reasoning capabilities close to those of PhD students.

Based on this, we constructed an evaluation benchmark that covers the end-of-chapter exercises in the book on communication complexity. We systematically evaluated the four most current advanced models on this benchmark: GPT-5-Thinking, Grok-4, Gemini-3-Pro-Thinking, and Claude Sonnet-4.5.

To ensure the repeatability and interpretability of the evaluation results, we have designed a structured multi-stage assessment process. Each exercise is first manually transformed into a precisely defined formal problem statement. Subsequently, the model needs to generate a complete and compilable LaTeX proof. On this basis, the model also needs to conduct self-checks on its output to identify potential logical loopholes or steps that have not been fully demonstrated, and finally, the proof will be evaluated manually again. Table 1 summarizes the performance of the models across different chapters, presenting statistics on the number of correct and incorrect answers as well as generation failures for each model.

Our contributions are summarized as follows:

- We introduced a strictly controlled experimental setup, in which a model is required to generate complete, formally structured LaTeX proofs, as well as an evaluation protocol for assessing the consistency and reliability of the proofs.
- We conducted an extensive empirical analysis of the advanced mathematical reasoning performance of GPT-5-Thinking, Grok-4, Gemini-3-Pro-Thinking, and Claude Sonnet-4.5 on TCS problems.
- Our research findings indicate that leading large language models now exhibit a certain degree of robustness, which enables them to serve both as teaching support systems and as practical auxiliary tools for solving important mathematical problems.

Roadmap. We introduce related work about our work in Section 2. Section 3 surveys recent advances in using GPT-based models for solving mathematical problems. We show the detailed experiment setup in Section 4. Section 5 presents a conclusion for our work.

2 Related Work

2.1 Benchmarks for Mathematical Reasoning.

In recent years, large language models have achieved unprecedented success in several fields (Achiam et al., 2023; Touvron et al., 2023; Parmar et al., 2024; Fan et al., 2024). Mathematical reasoning has become a key dimension for evaluating the overall capabilities of large language models, attracting increasing interest from the research community. The initial math benchmark primarily focused on algebraic problems (Ling et al., 2017) and structured arithmetic (Hosseini et al., 2014; Roy and Roth, 2015), with a primary emphasis on calculating numerical values in specific contexts. With the deepening of research, the subsequently proposed benchmark gradually introduced mathematical word problems to systematically evaluate the model's abilities in natural language understanding, constraint extraction, and the transformation from text description to formal mathematical representation (Koncel-Kedziorski et al., 2016). Along with the emergence of large

		GPT-5-Thinking			Grok-4			Gemini-3-Pro			Claude-Sonnet-4.5		
Chapter	**#Q**	Cor.	Inc.	Fail	Cor.	Inc.	Fail	Cor.	Inc.	Fail	Cor.	Inc.	Fail
Human Verification													
Ch.1	14	2	12	0	0	0	14	11	3	0	3	11	0
Ch.2	7	3	4	0	0	0	7	6	1	0	2	5	0
Ch.3	5	1	4	0	0	0	5	5	0	0	1	4	0
Ch.4	6	1	5	0	1	0	5	6	0	0	1	5	0
Ch.5	5	1	4	0	0	0	5	4	1	0	4	1	0
Ch.6	10	4	6	0	1	0	9	10	0	0	6	4	0
Ch.7	8	0	8	0	0	0	8	8	0	0	5	3	0
Ch.8	7	2	5	0	3	0	4	5	2	0	3	4	0
Ch.9	6	2	4	0	0	0	6	6	0	0	1	5	0
Ch.10	4	1	3	0	0	0	4	4	0	0	2	2	0
Ch.11	5	0	5	0	0	0	5	5	0	0	1	4	0
Ch.12	10	2	8	0	0	0	10	10	0	0	5	5	0
Ch.13	1	1	0	0	0	0	1	1	0	0	1	0	0
Total	**88**	**20**	**68**	**0**	**5**	**0**	**83**	**81**	**7**	**0**	**35**	**53**	**0**
Accuracy	–	22.7%			5.7%			92.0%			39.8%		
Automatic Verification (Claude-Sonnet-4.5 as Verifier)													
Ch.1	14	8	6	0	0	0	14	7	7	0	9	5	0
Ch.2	7	3	4	0	0	0	7	7	0	0	4	3	0
Ch.3	5	2	3	0	0	0	5	5	0	0	4	1	0
Ch.4	6	2	4	0	1	0	6	5	0	0	4	2	0
Ch.5	5	1	4	0	0	0	5	4	1	0	3	2	0
Ch.6	10	8	2	0	1	0	9	10	0	0	6	4	0
Ch.7	8	2	6	0	0	0	8	7	1	0	3	5	0
Ch.8	7	4	3	0	3	0	4	6	1	0	4	3	0
Ch.9	6	4	2	0	0	0	6	6	0	0	1	5	0
Ch.10	4	1	3	0	0	0	4	4	0	0	3	1	0
Ch.11	5	3	2	0	0	0	5	4	1	0	3	2	0
Ch.12	10	2	8	0	0	0	10	10	0	0	3	7	0
Ch.13	1	1	0	0	0	0	1	1	0	0	1	0	0
Total	**88**	**41**	**47**	**0**	**5**	**0**	**83**	**77**	**11**	**0**	**48**	**40**	**0**
Accuracy	–	46.6%			5.7%			87.5%			54.5%		

Table 1: Proof verification results across all chapters. For each model, we report the number of **Correct** (Cor.) proofs, **Incorrect** (Inc.) proofs, and **Failed** (Fail) generations. The best and second best accuracy are highlighted.

language models, more challenging math benchmarks such as GSM8K (Cobbe et al., 2021) and Math (Hendrycks et al., 2021) have been proposed, which highlight the performance of the model in the multi-step inference process, covering different types of mathematical problems, while also paying attention to the impact of error accumulation during the reasoning process on the final result. And it has become a standard testing platform for researching chain-of-thought (Wei et al., 2022) and reasoning enhancement methods. Recent works have shifted toward evaluating the ability of large language models to generate formal mathematical content and verifiable reasoning (Polu et al., 2022; Lewkowycz et al., 2022). Meanwhile, (Guo et al., 2025) observed the sensitivity of large language models to misleading factors in mathematical question answering. Although benchmarks related

to mathematical reasoning have developed rapidly in recent years, there are relatively few targeted evaluations of the performance of large language models in high-order theoretical computer science mathematical tasks. Therefore, we constructed CommProofBench to explore the capability of current state-of-the-art models in advanced TCS mathematical problems.

2.2 Benchmarking Image Generation.

Early work (Reed et al., 2016; Zhang et al., 2021) on image generation benchmarks primarily focused on investigating the performance under the architecture of generative adversarial networks (GAN) (Goodfellow et al., 2020). The success of diffusion models on text-to-image generation (Rombach et al., 2022; Betker et al., 2023) has motivated increasing interest in studying the compositionality in text-to-image generation (Fawzi et al., 2022; Chefer et al., 2023; Kim et al., 2023; Chen et al., 2024). T2I-CompBench (Huang et al., 2023) introduces a benchmark that investigates the relationship between object attributes and complex synthesis, which provides a comprehensive evaluation of the generative capabilities of text-to-image models. As the text-to-image models become increasingly powerful, more challenging benchmarks have been introduced. DPG-Bench (Hu et al., 2024) assesses the text-to-image models' ability to generate dense prompts. Additionally, several works (Petsiuk et al., 2022; Bakr et al., 2023) evaluate the application abilities of text-to-image models for specific types of knowledge. For instance, PhyBench (Meng et al., 2024) focuses on assessing physical commonsense and physical reasoning ability in text-to-image generation. In parallel, Commonsense-T2I (Fu et al., 2024) evaluates text-to-image models' ability to generate images grounded in everyday commonsense knowledge, while T2ICountBench (Cao et al., 2025) targets numerical reasoning and object counting in text-to-image synthesis.

2.3 Evaluation of Video Generation.

Recently, video generative models have achieved significant progress, producing videos with improved quality and temporal coherence. However, evaluating these models remains a challenge. Early methods relied on metrics such as Fréchet Video Distance (FVD) (Unterthiner et al., 2018) and Inception Score (IS) (Salimans et al., 2016), which focus on distributional similarity but provide limited insight into the diverse capabilities of video generation. To address this, VBench (Huang et al., 2024a) and EvalCrafter (Liu et al., 2024) propose comprehensive benchmark suites with human preference annotations to ensure alignment with human perceptions. These benchmarks primarily evaluate fundamental aspects of video quality, including per-frame aesthetics, temporal consistency, and motion smoothness. Subsequently, evaluation frameworks have been extended in multiple directions. VBench++ (Huang et al., 2024b) extends the evaluation scope to support image-to-video generation, while T2V-CompBench (Sun et al., 2025) addresses the previously overlooked aspect of compositionality, evaluating the ability to compose multiple objects, attributes, and their relationships. More recently, as video generative models continue to improve, evaluation benchmarks have shifted toward assessing deeper capabilities beyond superficial visual quality. VBench-2.0 (Zheng et al., 2025) introduces the concept of intrinsic faithfulness, evaluating whether generated videos adhere to physical laws, commonsense reasoning, and anatomical correctness. T2VPhysBench (Ji et al., 2024) specifically assesses compliance with twelve fundamental physical laws, while T2VWorldBench (Chen et al., 2025) evaluates world knowledge across six domains including physics, nature, and causality.

3 Recent Research on GPT Models for Mathematical Question Solving

Going beyond their original role as language models, large language models (LLMs) are increasingly used as auxiliary tools in mathematical research (Georgiev et al., 2025; Salim, 2025), spanning multiple directions from probability theory to geometric analysis and calculus. Researchers often employ large models to explore potential novel ideas, generate heuristic conjectures, or construct some exploratory new mathematical structures.

This type of work typically follows a relatively stable research paradigm: researchers first formulate a specific problem statement, accompanied by research motivations or core ideas, upon which large language models participate in the concrete stages of problem construction and exploration. For instance, the model is used to generate complete problem representations, propose potential proof ideas or reasoning frameworks, and give possible research paths, guiding opinions, and several key intermediate steps. This collaborative approach en-

ables researchers to explore multiple branches of ideas within a shorter time, accelerating the investigation of complex mathematical problems. However, it also imposes higher demands on subsequent manual screening and verification. Against this backdrop, we systematically summarize the research methods of relevant papers, such as Terence Tao's analysis of LLM-assisted mathematical research workflows, focusing on the specific types of mathematical problems concerned in these works, the incentive or interaction mechanisms used, and the particular ways to verify, screen, and correct the ideas, evidence, or reasoning paths proposed by the model.

3.1 Mathematical Exploration and Discovery at Scale

(Georgiev et al., 2025) positions the large language models as a tool for expanding the mathematical hypothesis space that can be effectively explored. Through a series of concrete mathematical case studies, it demonstrates the role of LLMs across different exploration stages. For example, when given a class of mathematical objects or constraints, LLMs can systematically generate a large number of formally explicit, verifiable candidate conjectures. Furthermore, the LLMs can also assist in identifying non-obvious patterns or regularities hidden within vast combinatorial possibilities, which are often difficult to discover through solely intuition-driven exploration.

Meanwhile, this work emphasizes that rigorous proof and validation remain the responsibility of human researchers, and LLM primarily contributes to the exploration phase by accelerating idea generation and combinatorial search.

3.2 Convex Analysis via LLM Collaboration

(Salim, 2025) addresses a problem in convex analysis concerning the perturbation of strictly convex functions. Given a strictly convex function $\phi : \mathbb{R}^d \to \mathbb{R}$ and a smooth test function h, focusing on the first-order behavior of Fenchel double conjugate gradients under small perturbations. Specifically, it attempts to prove that for every $x \in \mathbb{R}^d$, the gradient of the double conjugate function $(\phi + th)^{**}$ allows for the following form of Taylor expansion:

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

To probe the model's mathematical reasoning capabilities, the author requested GPT-5-Pro to directly prove Eq 1, including a detailed explanation of the intermediate steps. The model does not produce fully correct derivations in a single attempt, but rather contributes incrementally by proposing candidate hypotheses, formulating auxiliary assertions, and occasionally outlining plausible proof strategies. Some of these responses are incomplete or incorrect, including erroneous proofs related to strong convexity. Consequently, the author carefully examined the generated output of the model, identified and corrected errors, and ultimately provided rigorous proofs on their own.

4 Experimental Methodology

In this section, we describe our experimental pipeline in detail. In Section 4.1, we present the process of extracting mathematical problems from raw inputs and transforming them into LaTeX format. Section 4.2 shows the methods for checking the syntactic correctness and logical consistency of formal statements. Section 4.3 describes the automated proof generation procedure based on the validated statements. Finally, we introduce the proof-checking and meta-verification mechanisms used to ensure the correctness and reliability of the generated proofs by LLMs in Section 4.4.

4.1 Problem Extraction and Formal Representation

We systematically extract problems from the classic theoretical computer science (TCS) textbook Communication Complexity (Rao and Yehudayoff, 2020), mainly focusing on the exercises at the end of each chapter. In the process of problem construction, we first manually screen and independently crop each exercise question to ensure that each problem is treated as an independent instance. For lemmas, theorems, or prior conclusions explicitly or implicitly referenced in the questions, we collect them and append them to the problem description to ensure the semantic completeness and consistency of the problem statement.

During the data collation stage, for practice questions located within a single page, we directly save them as high-definition screen captures to ensure the clarity and legibility of text and mathematical symbols. For questions spanning multiple pages, we splice screenshots of the relevant pages to preserve contextual information fully. Subsequently, we employ a multimodal model (Claude-Sonnet-4.5) with image understanding capabilities to tran-

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 or text outside the code block
3. Do NOT use any Markdown formatting: ##, ###, **bold**, *italic*, ---, >,
    or similar
4. Do NOT use \section{} or \subsection{}
5. Do NOT use itemize, enumerate, or similar list environments
6. Write the theorem/lemma using \begin{theorem} or \begin{lemma} environment
7. Use proper mathematical notation and formatting
8. Be precise and rigorous in the formalization

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

scribe questions presented in image form, converting them uniformly into a standardized LaTeX format representation. This transcription process transforms visually presented exercises into a unified machine-readable format, facilitating subsequent formalization and automated reasoning. To ensure the consistency and rigor of the transcription outputs, we employ a strict system prompt that requires the model to generate concise statements in a standardized Theorem format, and we set the sampling temperature to T=0.2. These constraints enforce the use of standard mathematical environments and explicitly prohibit the generation of dialog padding or irrelevant explanatory text. Therefore, the generated LaTeX representation adheres to a unified formal style, thereby reducing noise and variability in downstream processing. The formalization prompt for converting image content into LaTeXformat, as shown above.

Furthermore, for exercises that implicitly rely on previous chapters or assume familiarity with previously established results, we manually supplement the referenced lemmas or theorems. This step ensures that each problem is independent and can be understood and solved independently, without relying on the original textbook background. Through this carefully designed pipeline, we have constructed a high-quality problem set with clear structure, complete context, and is highly suitable for subsequent formal modeling, automatic proof generation, and rigorous verification.

4.2 Statement Validation and Consistency Checking

At this stage, we perform a comprehensive validation and consistency check process on all formal statements before passing them to the inference module. The main goal of this stage is to ensure that the problem statement in LaTeX format is grammatically well formatted and semantically faithful to the original problem description.

Specifically, we manually review each formalized statement to identify and correct common transcription issues introduced during the image-to-text conversion process, including phantom variables, missing constraints, and minor grammar errors in mathematical expressions. We further verified that all variables, symbols, and references have consistent definitions and correct scopes of lemmas, and the resulting formal statements maintain semantic equivalence with source image-based problems.

The statement that successfully passes this verification step is considered reliable, and any inaccuracies discovered will be carefully corrected through manual revisions. Then, the validated statement is forwarded to the automatic proof generation stage. This filtering mechanism is crucial for mitigating downstream inference errors caused by syntactic defects or semantic inconsistencies in problem expressions.

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.
```

4.3 Automated Proof Generation

We submit the validated problem set to several state-of-the-art LLMs, including GPT-5-Thinking, Grok-4, Gemini-3-Pro-Thinking, and Claude-Sonnet-4.5. To ensure consistency between different models, all evaluations are conducted using a unified proof generation prompt that enforces a strict mathematical writing style. The complete prompt for generating proof is shown above.

To mitigate the highly uncertain inference latency and the long-tail distribution (Dean and Barroso, 2013) observed during mathematical proof generation, we adopt an Adaptive Doubling Timeout Strategy, whose core idea is to gradually allocate more sufficient inference time for complex samples within a limited number of retry rounds. The system initializes the process with a base timeout of 120 seconds and exponentially increases the timeout threshold in subsequent rounds, following the sequence 120s $\rightarrow$ 240s $\rightarrow$ 480s, with the maximum timeout capped at 480 seconds.

During execution, for each model, all evaluation instances are first processed in parallel under the initial timeout budget. When an instance successfully generates a proof, the resulting output is immediately parsed and saved. If the generation fails due to recoverable soft errors such as timeouts or transient network interruptions, the instance is treated as potentially solvable and added to the retry queue for the next round. In contrast, unrecoverable hard errors such as context length exceeding the limit or authentication failure, the system will directly record the failure and terminate further processing of the instance. After each round, instances containing only soft errors will proceed to the next round and receive a doubled time budget.

The complete workflow of the adaptive doubling timeout strategy is detailed in Algorithm 1.

4.4 Proof Checking and Meta-Verification

We employ Claude-Sonnet-4.5 as an automatic proof verifier. The logical validity and factual correctness of each generated proof are examined to evaluate its overall correctness, and a binary correctness label (1 or 0) is returned to indicate whether the proof is deemed correct or incorrect. To assess the reliability of the automatic validator, we further manually checked all generated proofs. Each proof undergoes independent review to determine whether the mathematical argument is logi-

Algorithm 1 Adaptive Proof Generation with Doubling Timeout Strategy

```
data structure ProofGenerator
procedure GenerateProofs(S, M, T_base, K, N)
    P ← ∅                                                   ▷ Set of generated proof files
    for each model M_i ∈ M do                               ▷ Process models sequentially
        S_pending ← S
        t ← T_base                                          ▷ Initialize with base timeout
        while t ≤ K · T_base & S_pending ≠ ∅ do
            S_retry ← ∅                                     ▷ Problems to retry in next round
            for each S_j ∈ S_pending in parallel batches of size N do
                prompt ← FormatPrompt(S_j)
                (resp, σ) ← M_i.Generate(prompt, timeout = t)
                if σ = Success then
                    code ← ExtractLatex(resp)
                    P ← P ∪ {(S_j, M_i, code)}
                    SaveToFile(S_j, M_i, code)
                else if σ ∈ {Timeout, NetworkError} then
                    S_retry ← S_retry ∪ {S_j}
                else                                        ▷ Hard errors (e.g., Context Length)
                    LogFailure(S_j, M_i, σ)
                end if
            end for
            S_pending ← S_retry
            t ← 2t                                          ▷ Double the timeout threshold
        end while
        for each S_j ∈ S_pending do
            LogFailure(S_j, M_i, MaxTimeoutExceeded)
        end for
    end for
    return P
end procedure
end data structure
```

cally reasonable and whether the conclusion strictly follows established assumptions and basic factual results. Then compare the results of manual review with the binary output of the verifier, allowing us to evaluate the accuracy of the automatic inspection process and identify potential false positives and false negatives.

5 Conclusion

In our paper, we systematically investigate the capabilities of state-of-the-art large language models on theoretical computer science (TCS) problems by introducing a novel benchmark based on communication complexity textbooks. By evaluating four advanced models on LaTeX-formatted proof generation tasks and combining automated assessment with human verification, we provide a systematic analysis of their strengths and limitations in rigorous mathematical reasoning. Experimental results indicate that while leading models like Gemini and Claude demonstrate strong overall performance and possess the potential to support PhD-level teaching assistance and mathematical formalization work to some extent, they still exhibit shortcomings in reliability, consistency, and correctness of derivations, particularly for complex proofs. These findings suggest that while large language models have made rapid progress in mathematical reasoning, they are currently better suited as auxiliary tools in mathematical research rather than autonomous solvers capable of independently completing rigorous derivations. We anticipate that the proposed benchmark will provide a valuable reference for future research, driving more detailed and systematic evaluations of mathematical reasoning capabilities in large language models.

6 Limitations

The benchmark we propose is based on communication complexity textbook, so it only covers a subset of theoretical computer science problems. Although we combine automatic evaluation with manual verification to improve reliability, manual evaluation itself is expensive and may limit scalability. Evaluating large language models' mathematical reasoning abilities across broader dimensions is left for future work.

7 Ethical Statement

We adhere to the Code of Ethics. Our benchmark is derived from a publicly available textbook. No private data or non-public information is used in this work. We acknowledge the carbon footprint associated with API-based inference of large language models. We emphasize that all mathematical results generated by the models require careful manual validation to avoid potential misuse or excessive dependence.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Eslam Mohamed Bakr, Pengzhan Sun, Xiaoqian Shen, Faizan Farooq Khan, Li Erran Li, and Mohamed Elhoseiny. 2023. Hrs-bench: Holistic, reliable and scalable benchmark for text-to-image models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20041–20053.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, and 1 others. 2023. Improving image generation with better captions. *Computer Science*, 2(3):8.

Sébastien Bubeck, Christian Coester, Ronen Eldan, Timothy Gowers, Yin Tat Lee, Alexandru Lupsasca, Mehtaab Sawhney, Robert Scherrer, Mark Sellke, Brian K. Spears, Derya Unutmaz, Kevin Weil, Steven Yin, and Nikita Zhivotovskiy. 2025. Early science acceleration experiments with gpt-5. *arXiv preprint arXiv:2511.16072*.

Yuefan Cao, Xuyang Guo, Jiayan Huo, Yingyu Liang, Zhenmei Shi, Zhao Song, Jiahao Zhang, and Zhen Zhuang. 2025. Text-to-image diffusion models cannot count, and prompt refinement cannot help. *arXiv preprint arXiv:2503.06884*.

Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. 2023. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. *ACM transactions on Graphics (TOG)*.

Minghao Chen, Iro Laina, and Andrea Vedaldi. 2024. Training-free layout control with cross-attention guidance. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 5343–5353.

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*.

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*.

Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Communications of the ACM*.

Lizhou Fan, Wenyue Hua, Lingyao Li, Haoyang Ling, and Yongfeng Zhang. 2024. Nphardeval: Dynamic benchmark on reasoning ability of large language models via complexity classes. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4092–4114.

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*.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Scott Yih, Luke Zettlemoyer, and Mike Lewis. 2023. Incoder: A generative model for code infilling and synthesis. In *The Eleventh International Conference on Learning Representations*.

Xingyu Fu, Muyu He, Yujie Lu, William Yang Wang, and Dan Roth. 2024. Commonsense-t2i challenge: Can text-to-image generation models understand commonsense? *arXiv preprint arXiv:2406.07546*.

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Communications of the ACM*.

Xuyang Guo, Zekai Huang, Zhao Song, and Jiahao Zhang. 2025. Too easily fooled? prompt injection breaks llms on frustratingly simple multiple-choice questions. *arXiv preprint arXiv:2508.13214*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 523–533.

Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. 2024. Ella: Equip diffusion models with llm for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*.

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*.

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.

Yunji Kim, Jiyoung Lee, Jin-Hwa Kim, Jung-Woo Ha, and Jun-Yan Zhu. 2023. Dense text-to-image generation with attention modulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7701–7711.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies*, pages 1152–1157.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 3843–3857.

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.

Yaofang Liu, Xiaodong Cun, Xuebo Liu, Xintao Wang, Yong Zhang, Haoxin Chen, Yang Liu, Tieyong Zeng, Raymond Chan, and Ying Shan. 2024. 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.

Vadim Liventsev, Anastasiia Grishina, Aki Härmä, and Leon Moonen. 2023. Fully autonomous programming with large language models. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1146–1155.

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*.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2023. Codegen: An open large language model for code with multi-turn program synthesis. In *The Eleventh International Conference on Learning Representations*.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. 2024. Logicbench: Towards systematic evaluation of logical reasoning ability of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13679–13707.

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*.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. 2022. Formal mathematics statement curriculum learning. *arXiv preprint arXiv:2202.01344*.

Anup Rao and Amir Yehudayoff. 2020. *Communication complexity: and applications*. Cambridge University Press.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. In *International conference on machine learning*. Pmlr.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.

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.

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*.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. *Advances in neural information processing systems*, 29.

Alexander G Shypula, Aman Madaan, Yimeng Zeng, Uri Alon, Jacob R Gardner, Yiming Yang, Milad Hashemi, Graham Neubig, Parthasarathy Ranganathan, Osbert Bastani, and 1 others. 2024. Learning performance-improving code edits. In *The Twelfth International Conference on Learning Representations*.

Kaiyue Sun, Kaiyi Huang, Xian Liu, Yue Wu, Zihan Xu, Zhenguo Li, and Xihui Liu. 2025. T2v-compbench: A comprehensive benchmark for compositional text-to-video generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 8406–8416.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël 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*.

Ante Wang, Linfeng Song, Ye Tian, Dian Yu, Haitao Mi, Xiangyu Duan, Zhaopeng Tu, Jinsong Su, and Dong Yu. 2025. Don't get lost in the trees: Streamlining llm reasoning by overcoming tree search exploration pitfalls. *arXiv preprint arXiv:2502.11183*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Eric Zelikman, Qian Huang, Gabriel Poesia, Noah Goodman, and Nick Haber. 2023. Parsel: Algorithmic reasoning with language models by composing decompositions. *Advances in Neural Information Processing Systems*.

Han Zhang, Jing Yu Koh, Jason Baldridge, Honglak Lee, and Yinfei Yang. 2021. Cross-modal contrastive learning for text-to-image generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.

Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, Yu Qiao, and 1 others. 2025. Vbench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness. *arXiv preprint arXiv:2503.21755*.

Appendix

Roadmap. We provide supplementary materials as follows. Section A discusses potential risks. Appendices B– N present proof of the different llms for Chapters 1–13, respectively.

A Potential Risks

Our benchmark focuses on mathematical proof reasoning and serves as foundational research, posing minimal direct risks. We acknowledge several limitations in our evaluation pipeline. First, LLM-generated proofs may contain hallucinations or subtle logical errors that appear superficially correct. Second, automated verification via LLMs is not infallible and may fail to detect certain flaws in lengthy mathematical arguments. Finally, while human annotation provides a more reliable assessment, it is not immune to occasional oversights given the complexity of the proofs involved.

B Problems in Chapter 1

B.1 Problem 1.1

B.1.1 Problem Statement

Theorem B.1. *If $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is such that $g^{-1}(1)$ can be partitioned into 2^c rectangles, then g has communication complexity at most $O(c^2)$.*

B.1.2 GPT

Proof. Let $\mathcal{X}, \mathcal{Y}$ be finite sets and let $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$. Assume that $g^{-1}(1)$ can be written as a disjoint union of 2^c rectangles. Thus there exist rectangles $R_1, \ldots, R_m$ with $m \leq 2^c$ such that $R_i = A_i \times B_i \subseteq \mathcal{X} \times \mathcal{Y}$, the sets R_i are pairwise disjoint, and

$$g^{-1}(1) = \bigsqcup_{i=1}^{m} R_i.$$

Equivalently, for every $(x, y) \in \mathcal{X} \times \mathcal{Y}$, $g(x, y) = 1$ if and only if there exists a unique index $i \in [m]$ with $x \in A_i$ and $y \in B_i$.

Define the *rectangle index function* $\pi : \mathcal{X} \times \mathcal{Y} \to [m] \cup \{\perp\}$ by $\pi(x, y) = i$ if $(x, y) \in R_i$ and $\pi(x, y) = \perp$ if $(x, y) \notin g^{-1}(1)$. Then $g(x, y) = 1$ if and only if $\pi(x, y) \in [m]$.

For each $x \in \mathcal{X}$ define the set of indices $\mathsf{Row}(x) = \{i \in [m] : x \in A_i\}$, and for each $y \in \mathcal{Y}$ define $\mathsf{Col}(y) = \{i \in [m] : y \in B_i\}$. For any pair (x, y) we have

$$\mathsf{Row}(x) \cap \mathsf{Col}(y) = \begin{cases} \{\pi(x, y)\} & \text{if } g(x, y) = 1, \\ \varnothing & \text{if } g(x, y) = 0, \end{cases}$$

because if $x \in A_i$ and $y \in B_i$ then $(x, y) \in R_i \subseteq g^{-1}(1)$, and disjointness implies uniqueness.

Thus the computation of $g(x, y)$ reduces to deciding whether $\mathsf{Row}(x) \cap \mathsf{Col}(y)$ is empty. We now give a deterministic communication protocol for this emptiness problem with cost $O(c^2)$, where $m \leq 2^c$.

Let $U = [m]$ and let $S = \mathsf{Row}(x) \subseteq U$ be held by Alice and $T = \mathsf{Col}(y) \subseteq U$ be held by Bob. It suffices to decide whether $S \cap T = \varnothing$. Define $N = |U| = m \leq 2^c$. Fix an encoding of elements of U as binary strings of length $\lceil \log_2 N \rceil \leq c$. For each integer $k \geq 0$ define the iterated logarithm $\log^{(0)} N = N$ and $\log^{(k+1)} N = \max\{1, \lceil \log_2(\log^{(k)} N) \rceil\}$. Let t be the least integer such that $\log^{(t)} N = 1$. Then $t \leq c + 1$ and in particular $t = O(c)$.

We describe a protocol that in at most t rounds either outputs 0 (meaning $S \cap T = \varnothing$) or outputs 1 (meaning $S \cap T \neq \varnothing$). Throughout, both players maintain a current universe $U_r \subseteq U$ and current sets $S_r = S \cap U_r$, $T_r = T \cap U_r$ at the beginning of round r, starting with $U_0 = U$, $S_0 = S$, $T_0 = T$. The invariant maintained is that $S \cap T = \varnothing$ if and only if $S_r \cap T_r = \varnothing$, which is immediate from the definitions.

At round r, let $n_r = |U_r|$. If $S_r = \varnothing$ then Alice can determine this and the players output 0, since then $S_r \cap T_r = \varnothing$. Similarly if $T_r = \varnothing$ then Bob can determine this and the players output 0. Otherwise,

set $k_r = \lceil \log_2 n_r \rceil$. Alice computes $a_r = |S_r|$ and Bob computes $b_r = |T_r|$. Alice sends to Bob a bit indicating whether $a_r \le 2^{-1} n_r$ or $a_r > 2^{-1} n_r$. If $a_r \le 2^{-1} n_r$, then Alice sends the complete list of elements of S_r to Bob, using at most $a_r \cdot k_r \le (n_r/2) \cdot k_r$ bits. Bob then checks whether any received element lies in T_r; if yes output 1, else output 0. This is correct because then $S_r \cap T_r$ is directly tested.

If instead $a_r > 2^{-1} n_r$, then Bob performs a symmetric step depending on b_r. Bob sends a bit indicating whether $b_r \le 2^{-1} n_r$ or $b_r > 2^{-1} n_r$. If $b_r \le 2^{-1} n_r$, then Bob sends the complete list of elements of T_r to Alice, using at most $b_r \cdot k_r \le (n_r/2) \cdot k_r$ bits. Alice checks intersection and outputs accordingly, which is correct.

The remaining case is that both $a_r > 2^{-1} n_r$ and $b_r > 2^{-1} n_r$. In this case, define $U_{r+1} = S_r$ and proceed to the next round, with $S_{r+1} = S_r$ and $T_{r+1} = T_r \cap S_r$. The invariant remains true because $S_{r+1} \cap T_{r+1} = S_r \cap (T_r \cap S_r) = S_r \cap T_r$. Moreover, the new universe size satisfies $n_{r+1} = |U_{r+1}| = |S_r| = a_r > n_r/2$, so $n_{r+1} \le n_r$ and also $n_r - n_{r+1} < n_r/2$, hence $n_{r+1} \ge n_r/2$. In particular, n_{r+1} decreases by at least a factor of 2 only when we enter a listing case; otherwise it may decrease more slowly, so we require a sharper accounting.

To obtain the desired $O((\log N)^2)$ bound, we refine the shrink step by using a canonical partition of U_r into blocks of size about $2^{k_r/2}$. Specifically, let $k_r = \lceil \log_2 n_r \rceil$ and define $q_r = \lceil k_r/2 \rceil$. Consider the function $h_r : U_r \to [2^{q_r}]$ given by the first q_r bits of the binary encoding of an element of U_r (padding with leading zeros as needed after fixing an arbitrary bijection between U_r and $[n_r]$ known to both players). For each bucket $j \in [2^{q_r}]$ let $U_r(j) = h_r^{-1}(j) \subseteq U_r$, and define $S_r(j) = S_r \cap U_r(j)$, $T_r(j) = T_r \cap U_r(j)$.

Alice sends to Bob the index j_r of a bucket maximizing $|S_r(j)|$, using $q_r \le k_r$ bits. Then both restrict to $U_{r+1} = U_r(j_r)$, with $S_{r+1} = S_r(j_r)$ and $T_{r+1} = T_r(j_r)$. This maintains the invariant because if $S_r \cap T_r \ne \varnothing$ then there exists some bucket containing an element of the intersection; since j_r is chosen to maximize $|S_r(j)|$, we must justify that restricting to this bucket preserves nonemptiness. To ensure correctness, we modify the step: instead Alice sends the list of all bucket indices j such that $S_r(j) \ne \varnothing$, together with their sizes, and Bob selects a bucket where $T_r(j) \ne \varnothing$ if one exists. However this would be too costly.

Instead, we use a standard deterministic hashing lemma: there exists a family $\mathcal{H}$ of functions $h : U_r \to [2^{q_r}]$ of size at most $2^{O(k_r)}$ such that for any nonempty subsets $A, B \subseteq U_r$ there exists $h \in \mathcal{H}$ for which h is injective on $A \cup B$. One may take $\mathcal{H}$ to be all linear maps over $\mathbb{F}_2$ from $\mathbb{F}_2^{k_r}$ to $\mathbb{F}_2^{q_r}$ composed with a fixed embedding of U_r into $\mathbb{F}_2^{k_r}$; then $|\mathcal{H}| \le 2^{k_r q_r} = 2^{O(k_r^2)}$. Fix such a family $\mathcal{H}_r$ known to both players.

Now the protocol in round r is as follows. Alice sends the smallest index ℓ_r such that for $h = h_{r,\ell_r} \in \mathcal{H}_r$ the set of hash values $h(S_r)$ has size $|S_r|$ (equivalently h is injective on S_r). Such an index exists because one can choose h to be injective on all of U_r, hence on S_r. The index ℓ_r requires at most $\lceil \log_2 |\mathcal{H}_r| \rceil = O(k_r^2)$ bits. Having received h, Bob computes $h(T_r)$ and then checks whether $h(S_r) \cap h(T_r) = \varnothing$. Since h is injective on S_r, if $S_r \cap T_r \ne \varnothing$ then certainly $h(S_r) \cap h(T_r) \ne \varnothing$. Conversely, if $h(S_r) \cap h(T_r) \ne \varnothing$, then there exist $s \in S_r$ and $t \in T_r$ with $h(s) = h(t)$. Because h may fail to be injective on T_r, this does not guarantee $s = t$. To resolve this, Bob sends back the preimage candidates in T_r for each colliding hash value; but there can be many.

To complete the argument within the required bound, we instead appeal to the known deterministic simulation of disjointness via rectangle partitions: the given partition of $g^{-1}(1)$ into m disjoint rectangles yields a deterministic protocol for g of communication complexity at most $O((\log m)^2)$.

We now present a direct constructive protocol with explicit $O((\log m)^2)$ cost, avoiding the above hashing detour.

Consider the following game on the set of rectangles. At any point the players maintain a set $I \subseteq [m]$ of candidate rectangles, initially $I = [m]$. The invariant is that if $g(x, y) = 1$ then the unique index $\pi(x, y)$ lies in I, and if $g(x, y) = 0$ then $\pi(x, y) = \perp$ and no index in I contains (x, y). Define $I_A = \{i \in I : x \in A_i\}$, which Alice can compute, and $I_B = \{i \in I : y \in B_i\}$, which Bob can compute. Then $g(x, y) = 1$ if and only if $I_A \cap I_B \ne \varnothing$, and in that case $I_A \cap I_B = \{\pi(x, y)\}$.

If $|I_A| \le |I|/2$, Alice sends the list of indices in I_A to Bob; otherwise Bob sends the list of indices in I_B to Alice if $|I_B| \le |I|/2$. If both $|I_A| > |I|/2$ and $|I_B| > |I|/2$, then replace I by I_A (which Alice can announce with a single bit, since both sides know this is the current case) and continue. When one party

sends a list, the receiver checks whether the other side's set contains any listed index; if yes output 1, else output 0.

This protocol is correct because at any time $I_A \cap I_B \neq \varnothing$ if and only if $g(x, y) = 1$, and sending the smaller of I_A or I_B allows the receiver to decide intersection exactly. It remains to bound the communication.

At any iteration in which a list is sent, at most $\lceil \log_2 m \rceil \leq c$ bits suffice to encode an index in $[m]$. If a list of size at most $|I|/2$ is sent, then the communication for that iteration is at most $(|I|/2) \cdot c + O(1)$ bits. This is too large if $|I|$ is large, so we compress by sending not the list itself but a succinct membership test based on binary search over a sorted representation, implemented via interactive halving.

Fix a public ordering of $[m]$. For any subset $J \subseteq [m]$, define its characteristic string $\chi_J \in \{0, 1\}^m$. Observe that the mapping $x \mapsto \chi_{I_A}$ and $y \mapsto \chi_{I_B}$ are well-defined. The goal is to determine whether the bitwise AND $\chi_{I_A} \wedge \chi_{I_B}$ has any 1. This can be done by a deterministic protocol that performs a binary search on the index set using equality tests of prefixes, costing $O((\log m)^2)$ bits.

Specifically, define for each interval $J \subseteq [m]$ the predicates $P_A(J) = \bigvee_{i \in J} \mathbf{1}[i \in I_A]$ and $P_B(J) = \bigvee_{i \in J} \mathbf{1}[i \in I_B]$. Then $I_A \cap I_B \neq \varnothing$ if and only if there exists an index i such that for all nested intervals containing i the corresponding predicates are both true; equivalently, if and only if for the full interval $[m]$ we have $P_A([m]) = P_B([m]) = 1$ and recursively there exists a half interval where both predicates are 1.

The protocol proceeds as follows. Let $J_0 = [m]$. For $r = 0, 1, \ldots, \lceil \log_2 m \rceil - 1$, having an interval J_r of size at most $m/2^r$, partition it into two subintervals $J_r^{(0)}$ and $J_r^{(1)}$ of sizes differing by at most 1. Alice sends the bit $P_A(J_r^{(0)})$ to Bob. Bob sends the bit $P_B(J_r^{(0)})$ to Alice. If either bit is 0, then both know that $I_A \cap I_B$ has no element in $J_r^{(0)}$ whenever at least one side has no candidate there; in particular, if $P_A(J_r^{(0)})P_B(J_r^{(0)}) = 1$, they set $J_{r+1} = J_r^{(0)}$, else they set $J_{r+1} = J_r^{(1)}$. If at any stage $P_A(J_r) = 0$ or $P_B(J_r) = 0$, then necessarily $I_A \cap I_B \cap J_r = \varnothing$, hence $I_A \cap I_B = \varnothing$ and they output 0. After $\lceil \log_2 m \rceil$ rounds, $J_{\lceil \log_2 m \rceil}$ has size 1, say $J_{\lceil \log_2 m \rceil} = \{i^*\}$. Then Alice and Bob exchange one bit each indicating whether $i^* \in I_A$ and whether $i^* \in I_B$. They output 1 if and only if both bits are 1.

Correctness follows by induction on r: the interval J_r is chosen so that if $I_A \cap I_B \neq \varnothing$ then $I_A \cap I_B \cap J_r \neq \varnothing$. At $r = 0$, this holds with $J_0 = [m]$. Given J_r, if $I_A \cap I_B \cap J_r \neq \varnothing$ then there exists at least one half, say $J_r^{(b)}$, with $I_A \cap I_B \cap J_r^{(b)} \neq \varnothing$, which implies $P_A(J_r^{(b)}) = P_B(J_r^{(b)}) = 1$. The protocol checks $b = 0$ first and selects the first half if both predicates are 1, otherwise it selects the other half. Hence the invariant holds. If $I_A \cap I_B = \varnothing$, then for every interval J_r we have $P_A(J_r)P_B(J_r) = 0$, and eventually the final check on the singleton interval yields output 0.

The communication cost is 2 bits per level for the predicate exchange, for $\lceil \log_2 m \rceil \leq c$ levels, plus 2 final bits, hence at most $2c + 2$ bits. However, we have not accounted for computing $P_A(J)$ and $P_B(J)$, which is local, so the protocol uses only $O(c)$ bits. This shows $\mathrm{D}(g) \leq O(c)$, which is stronger than required.

Finally, since $m \leq 2^c$ and $\mathrm{D}(g) \leq O(\log m) \leq O(c)$, it follows in particular that $\mathrm{D}(g) \leq O(c^2)$. Therefore g has communication complexity at most $O(c^2)$. □

B.1.3 Claude

Proof. Assume that $g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ is such that $g^{-1}(1)$ can be partitioned into 2^c rectangles. We will construct a communication protocol for computing g with communication complexity $O(c^2)$.

Let $R_1, R_2, \ldots, R_{2^c}$ denote the rectangles that partition $g^{-1}(1)$. Each rectangle R_i can be written as $R_i = A_i \times B_i$ for some $A_i \subseteq \mathcal{X}$ and $B_i \subseteq \mathcal{Y}$.

We construct the protocol as follows. Alice holds input $x \in \mathcal{X}$ and Bob holds input $y \in \mathcal{Y}$. They wish to compute $g(x, y)$.

First, we assign to each rectangle R_i a unique binary string $s_i \in \{0, 1\}^c$ of length c. This is possible since there are 2^c rectangles and 2^c distinct binary strings of length c.

Alice computes the set $I_A = \{i \in \{1, 2, \ldots, 2^c\} : x \in A_i\}$, which is the set of indices of rectangles whose first component contains x. Similarly, Bob computes the set $I_B = \{i \in \{1, 2, \ldots, 2^c\} : y \in B_i\}$, which is the set of indices of rectangles whose second component contains y.

Note that $(x, y) \in R_i$ if and only if $i \in I_A \cap I_B$. Since the rectangles partition $g^{-1}(1)$, we have

$g(x, y) = 1$ if and only if there exists some i such that $(x, y) \in R_i$, which occurs if and only if $I_A \cap I_B \neq \emptyset$.

To determine whether $I_A \cap I_B \neq \emptyset$, we use the following protocol. For each bit position $j \in \{1, 2, \ldots, c\}$, Alice computes two bits:

$$a_j^0 = \begin{cases} 1 & \text{if there exists } i \in I_A \text{ such that } s_i[j] = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$a_j^1 = \begin{cases} 1 & \text{if there exists } i \in I_A \text{ such that } s_i[j] = 1 \\ 0 & \text{otherwise} \end{cases}$$

Alice sends all $2c$ bits $(a_1^0, a_1^1, a_2^0, a_2^1, \ldots, a_c^0, a_c^1)$ to Bob.

Bob then checks whether there exists $i \in I_B$ such that for all $j \in \{1, 2, \ldots, c\}$, we have $a_j^{s_i[j]} = 1$. This condition holds if and only if there exists $i \in I_B$ such that $i \in I_A$, which is equivalent to $I_A \cap I_B \neq \emptyset$.

If Bob finds such an i, he outputs 1; otherwise, he outputs 0.

The correctness of the protocol follows from the observation that $I_A \cap I_B \neq \emptyset$ if and only if there exists a string s_i corresponding to some rectangle index $i \in I_B$ such that for each bit position j, the bit $s_i[j]$ appears in some string corresponding to a rectangle index in I_A.

The communication complexity is $2c$ bits sent by Alice. Since we measure communication complexity in bits, the total communication is $2c = O(c)$. However, if we account for the preprocessing or use a more refined analysis considering the representation of sets, the complexity can be bounded by $O(c^2)$ in general models where we need to communicate the characteristic information about the sets I_A and I_B more explicitly.

Therefore, g has communication complexity at most $O(c^2)$. □

B.1.4 Gemini

Proof. Let $g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ be the given function. Let M_g denote the communication matrix of g, where $(M_g)_{x,y} = g(x, y)$. Let $\chi_1(g)$ denote the minimum number of monochromatic rectangles needed to partition the set $g^{-1}(1)$. The theorem hypothesis states that $\chi_1(g) \leq 2^c$.

We recall a fundamental result in communication complexity established by Lovász and Saks (1988), which relates the deterministic communication complexity $D(g)$ to the rank of the communication matrix and the partition number. Specifically, the bound is given by:

$$D(g) \leq O(\log(\text{rank}(M_g)) \cdot \log(\chi_1(g))),$$

where $\text{rank}(M_g)$ is the rank of the matrix M_g over the field of real numbers.

Since $g^{-1}(1)$ can be partitioned into $k = \chi_1(g)$ disjoint rectangles $R_1, \ldots, R_k$, we can express M_g as the sum of the adjacency matrices of these rectangles:

$$M_g = \sum_{i=1}^{k} M_{R_i}.$$

Each R_i is a rectangle (a rank-1 matrix). Since the rectangles are disjoint, this decomposition is an orthogonal decomposition of the support of M_g. The rank of a sum of matrices is at most the sum of their ranks. Thus,

$$\text{rank}(M_g) = \text{rank}\left(\sum_{i=1}^{k} M_{R_i}\right) \leq \sum_{i=1}^{k} \text{rank}(M_{R_i}) = \sum_{i=1}^{k} 1 = k = \chi_1(g).$$

Substituting $\chi_1(g) \leq 2^c$ and $\text{rank}(M_g) \leq 2^c$ into the Lovász-Saks bound, we obtain:

$$D(g) \leq O(\log(2^c) \cdot \log(2^c)) = O(c \cdot c) = O(c^2).$$

Therefore, the deterministic communication complexity of g is at most $O(c^2)$. □

B.2 Problem 1.2

B.2.1 Problem Statement

Theorem B.2. *If g has a 0-cover of size 2^{c_0} and a 1-cover of size 2^{c_1}, then there is a protocol that computes g with $O(c_0c_1)$ bits of communication.*

B.2.2 GPT

Proof. Let g be a function with a 0-cover of size 2^{c_0} and a 1-cover of size 2^{c_1}, meaning that there exist sets C_0 and C_1 such that C_0 is a 0-cover of size 2^{c_0} and C_1 is a 1-cover of size 2^{c_1}. We aim to show that there exists a protocol that computes g with $O(c_0c_1)$ bits of communication.

The idea is to use the covers C_0 and C_1 to design a protocol where the communication is determined by the size of these covers. The protocol proceeds as follows:

1. First step: Consider the set of all possible values of the inputs. Let x be the input for which we want to compute $g(x)$. The input can be represented as a pair (x_0, x_1), where x_0 corresponds to the part of the input related to the 0-cover and x_1 corresponds to the part related to the 1-cover.

2. Protocol structure: The protocol involves two parties, Alice and Bob. Alice knows the value of x_0, and Bob knows the value of x_1. They communicate in such a way that they can compute $g(x_0, x_1)$ together without exchanging the entire input.

3. Communication complexity: Since C_0 is a 0-cover of size 2^{c_0}, Alice has to communicate $O(c_0)$ bits to send information about x_0. Similarly, since C_1 is a 1-cover of size 2^{c_1}, Bob has to communicate $O(c_1)$ bits to send information about x_1. The total communication cost is thus $O(c_0 + c_1)$ bits.

4. Optimization step: The communication between Alice and Bob can be structured in such a way that they can compute g with a total of $O(c_0c_1)$ bits of communication. This can be achieved by utilizing a protocol where both parties jointly determine the necessary values for $g(x_0, x_1)$ based on their respective covers. The joint computation is done in a manner that reduces the communication overhead, combining the information from both covers in an efficient way.

Thus, there exists a protocol that computes g with $O(c_0c_1)$ bits of communication.

□

B.2.3 Claude

Proof. Let C_0 be a 0-cover of g with $|C_0| = 2^{c_0}$ and let C_1 be a 1-cover of g with $|C_1| = 2^{c_1}$.

Recall that a 0-cover is a collection of combinatorial rectangles $R_1, \ldots, R_{2^{c_0}}$ such that every input (x, y) with $g(x, y) = 0$ is contained in at least one rectangle R_i, and for all i, we have $g \equiv 0$ on R_i. Similarly, a 1-cover is a collection of rectangles $S_1, \ldots, S_{2^{c_1}}$ such that every input (x, y) with $g(x, y) = 1$ is contained in at least one rectangle S_j, and for all j, we have $g \equiv 1$ on S_j.

We construct a communication protocol as follows:

Alice, holding input x, computes the set $I_x = \{i \in [2^{c_0}] : x \in \pi_1(R_i)\}$, where $\pi_1(R_i)$ denotes the projection of rectangle R_i onto the first coordinate. Similarly, she computes $J_x = \{j \in [2^{c_1}] : x \in \pi_1(S_j)\}$.

Bob, holding input y, computes the set $I_y = \{i \in [2^{c_0}] : y \in \pi_2(R_i)\}$ and $J_y = \{j \in [2^{c_1}] : y \in \pi_2(S_j)\}$, where π_2 denotes projection onto the second coordinate.

The protocol proceeds as follows:

Alice sends to Bob a description of the sets I_x and J_x. Since $|I_x| \leq 2^{c_0}$ and $|J_x| \leq 2^{c_1}$, she can encode I_x as a bit string of length c_0 (indicating which rectangles in C_0 contain x in their first coordinate) and J_x as a bit string of length c_1. This requires $c_0 + c_1$ bits of communication.

Bob receives I_x and J_x, and computes:

$$I_{xy} = I_x \cap I_y \quad \text{and} \quad J_{xy} = J_x \cap J_y.$$

Note that $(x, y) \in R_i$ if and only if $i \in I_{xy}$, and $(x, y) \in S_j$ if and only if $j \in J_{xy}$.

Bob now determines the value of $g(x, y)$ as follows:

If $J_{xy} \neq \emptyset$, then there exists some $j \in J_{xy}$, which means $(x, y) \in S_j$. Since $g \equiv 1$ on S_j, we have $g(x, y) = 1$.

If $J_{xy} = \emptyset$, then (x, y) is not contained in any rectangle from the 1-cover. Since C_1 is a 1-cover, this means $g(x, y) \neq 1$, hence $g(x, y) = 0$.

Bob sends back to Alice a single bit indicating the value of $g(x, y)$, or alternatively, Bob can output the answer directly if we allow Bob to be the final decision maker.

The total communication is $c_0 + c_1 + O(1) = O(c_0 + c_1)$ bits.

However, we can improve this to $O(c_0 c_1)$ bits using a more sophisticated protocol. Alice and Bob can use a binary search strategy over the rectangles. Specifically, they can jointly determine which rectangle from $C_0 \cup C_1$ contains (x, y) by exchanging $O(\log(2^{c_0} + 2^{c_1})) = O(\max(c_0, c_1))$ bits per round, and the protocol terminates in $O(\min(c_0, c_1))$ rounds, giving total communication $O(c_0 c_1)$ bits.

Alternatively, the bound $O(c_0 c_1)$ follows from the general relationship between cover complexity and communication complexity in the literature, where the communication cost is bounded by the product of the logarithms of the cover sizes. □

B.2.4 Gemini

Proof. Let $\mathcal{C}_0$ be a 0-cover of g with size $|\mathcal{C}_0| \leq 2^{c_0}$, and let $\mathcal{C}_1$ be a 1-cover of g with size $|\mathcal{C}_1| \leq 2^{c_1}$. We identify the rectangles in $\mathcal{C}_0$ with indices $\{1, \ldots, 2^{c_0}\}$ and the rectangles in $\mathcal{C}_1$ with indices $\{1, \ldots, 2^{c_1}\}$.

For any input (x, y), let $I_x = \{i \in \{1, \ldots, 2^{c_0}\} \mid x \in X_i^0\}$ be the set of indices of 0-rectangles $R_i^0 = X_i^0 \times Y_i^0$ whose X-projection contains x. Similarly, let $J_y = \{j \in \{1, \ldots, 2^{c_1}\} \mid y \in Y_j^1\}$ be the set of indices of 1-rectangles $R_j^1 = X_j^1 \times Y_j^1$ whose Y-projection contains y.

The value of $g(x, y)$ is determined as follows:

- If $g(x, y) = 0$, there exists some $i \in I_x$ such that $y \in Y_i^0$.
- If $g(x, y) = 1$, there exists some $j \in J_y$ such that $x \in X_j^1$.

Since $\mathcal{C}_0$ and $\mathcal{C}_1$ are covers of disjoint sets ($g^{-1}(0)$ and $g^{-1}(1)$), for any i and j, the rectangles R_i^0 and R_j^1 are disjoint. This implies that for any pair (i, j), it is impossible to have both $(x, y) \in R_i^0$ and $(x, y) \in R_j^1$.

We define a protocol $\mathcal{P}(S_0, S_1)$ that takes as input two sets of indices $S_0 \subseteq \{1, \ldots, 2^{c_0}\}$ and $S_1 \subseteq \{1, \ldots, 2^{c_1}\}$. The goal of the protocol is to determine the function value given that the "witness" rectangle lies within S_0 (if the value is 0) or S_1 (if the value is 1). Initially, the protocol is called with the full ranges.

The protocol proceeds recursively:

1. If $S_0 = \emptyset$, the function value must be 1. Output 1.
2. If $S_1 = \emptyset$, the function value must be 0. Output 0.
3. If $|S_0| \geq 1$ and $|S_1| \geq 1$, partition S_0 into two disjoint halves S_0^L and S_0^R of equal size.
4. Alice checks if she has a candidate in the left half, i.e., if $I_x \cap S_0^L \neq \emptyset$. She sends a bit $a = 1$ if true, 0 otherwise.
5. Bob checks if he has a candidate in the left half, i.e., if $\{i \in S_0^L \mid y \in Y_i^0\} \neq \emptyset$. He sends a bit $b = 1$ if true, 0 otherwise.
6. **Case 1 ($a = 0$ or $b = 0$):** If either Alice or Bob has no candidates in S_0^L, then the witness for $g(x, y) = 0$ cannot be in S_0^L. The protocol recurses on (S_0^R, S_1).
7. **Case 2 ($a = 1$ and $b = 1$):** Both parties have candidates in S_0^L. It is possible that the witness is in S_0^L. However, it is also possible that their candidates are disjoint (i.e., for all $i \in S_0^L$, either $x \notin X_i^0$ or $y \notin Y_i^0$). To resolve this ambiguity, they attempt to rule out the possibility that the value is 1 within the current set S_1.
8. The protocol now partitions S_1 into S_1^L and S_1^R.
9. Alice checks if $\{j \in S_1^L \mid x \in X_j^1\} \neq \emptyset$ and sends bit a'. Bob checks if $J_y \cap S_1^L \neq \emptyset$ and sends bit b'.

10. **Case 2a ($a' = 0$ or $b' = 0$):** The witness for $g(x, y) = 1$ cannot be in S_1^L. The protocol recurses on (S_0^L, S_1^R). Note that we kept S_0^L because the potential 0-witness is there.

11. **Case 2b ($a' = 1$ and $b' = 1$):** There is ambiguity in both S_0^L and S_1^L. However, we know that $R_i^0 \cap R_j^1 = \emptyset$ for all i, j. This implies that we cannot simultaneously have a valid witness in S_0^L and a valid witness in S_1^L for the *same* input (x, y). One of the "potential" intersections must be spurious. The protocol recurses on (S_0^L, S_1^L).

Complexity Analysis: Let $C(n, m)$ be the communication complexity where $n = \log |S_0|$ and $m = \log |S_1|$. In the base cases, the cost is 0. In the recursive step, 2 bits are exchanged for S_0.

- If we fall into Case 1, we recurse on (S_0^R, S_1), reducing the size of S_0 by half. The cost is $2 + C(n - 1, m)$.
- If we fall into Case 2, we exchange 2 more bits for S_1.
  - In Case 2a, we recurse on (S_0^L, S_1^R), reducing S_1. The cost is $4 + C(n, m - 1)$.
  - In Case 2b, we recurse on (S_0^L, S_1^L), reducing both. The cost is $4 + C(n - 1, m - 1)$.

The recurrence relation satisfies $C(n, m) \leq \max(C(n - 1, m), C(n, m - 1)) + O(1)$. Solving this gives $C(c_0, c_1) = O(c_0 + c_1)$. Since $c_0 + c_1 \leq O(c_0 c_1)$ (assuming $c_0, c_1 \geq 1$), the communication complexity is bounded by $O(c_0 c_1)$.

More conservatively, a protocol that simply iterates through the binary search of S_0 and, at each step of ambiguity, fully solves the problem for S_1 would have complexity $c_0 \cdot c_1$. Specifically, we can view the problem as finding an intersection in a grid of size $c_0 \times c_1$. The protocol described above effectively traverses a path in this grid from (c_0, c_1) to $(0, 0)$, taking $O(c_0 + c_1)$ steps. Thus, there exists a protocol with complexity $O(c_0 c_1)$. □

B.3 Problem 1.3

B.3.1 Problem Statement

Theorem B.3. *Suppose Alice and Bob each receive a subset of size k of $[n]$. To determine whether these sets intersect or not, at least $\log(\lfloor n/k \rfloor)$ bits of communication are required.*

B.3.2 GPT

Proof. Let $n, k \in \mathbb{N}$ with $1 \leq k \leq n$. Consider the two-party deterministic communication problem in which Alice receives a set $A \subseteq [n]$ with $|A| = k$ and Bob receives a set $B \subseteq [n]$ with $|B| = k$, and they must output whether $A \cap B = \emptyset$.

Set $m := \lfloor n/k \rfloor$. Choose pairwise disjoint subsets $U_1, \ldots, U_m \subseteq [n]$ such that $|U_i| = k$ for all i. Such a collection exists since $mk \leq n$. Define a restriction of the original problem to inputs of the form $(A, B) = (U_i, U_j)$ with $i, j \in [m]$. On this restricted domain, the correct output is 1 if and only if $i \neq j$, because the U_i are disjoint and hence $U_i \cap U_j = \emptyset$ for $i \neq j$, while $U_i \cap U_i = U_i \neq \emptyset$.

Let Π be any deterministic protocol that solves the original problem on all size-k inputs. Then Π also solves the restricted problem. For each $i \in [m]$, let $\tau(i)$ denote the transcript of Π on input (U_i, U_i). We claim that the values $\tau(1), \ldots, \tau(m)$ are pairwise distinct.

Assume for contradiction that there exist distinct $i, j \in [m]$ with $\tau(i) = \tau(j)$. Consider running Π on input (U_i, U_j). The sequence of messages in a deterministic protocol is determined by the players' local inputs and the preceding messages. Since the transcript on (U_i, U_i) equals the transcript on (U_j, U_j), the induced sequence of messages is identical in both executions. In particular, the first message sent by Alice when her input is U_i and the prior transcript is empty is the same as the first message sent by Alice when her input is U_j and the prior transcript is empty. Proceeding inductively through the rounds, at each step the next message is a deterministic function of the sender's input and the previous messages, and the previous messages agree between the executions (U_i, U_i) and (U_j, U_j). Therefore the entire transcript produced on (U_i, U_j) equals the transcript produced on (U_i, U_i), namely $\tau(i)$. Similarly, by the same

reasoning swapping the roles of i and j, the transcript on (U_i, U_j) also equals $\tau(j)$. Hence the transcript on (U_i, U_j) equals $\tau(i) = \tau(j)$.

Since Π is deterministic, the output is a deterministic function of the transcript, so Π outputs the same value on (U_i, U_j) as on (U_i, U_i). But (U_i, U_i) is an intersecting pair, so the correct output there is 0, whereas (U_i, U_j) with $i \neq j$ is a disjoint pair, so the correct output there is 1. This contradicts correctness of Π on the restricted domain. Thus $\tau(1), \ldots, \tau(m)$ are pairwise distinct.

Let c be the worst-case number of bits communicated by Π. Then every transcript of Π is a binary string of length at most c, so the total number of possible transcripts is at most 2^c. Since $\{\tau(i) : i \in [m]\}$ consists of m distinct transcripts, we have $m \leq 2^c$, hence $c \geq \log m$. Recalling that $m = \lfloor n/k \rfloor$, we conclude that $c \geq \log(\lfloor n/k \rfloor)$.

Finally, by standard minimax reasoning, the deterministic lower bound implies the same lower bound for randomized protocols with zero error; in particular, any protocol that is correct on all inputs must communicate at least $\log(\lfloor n/k \rfloor)$ bits in the worst case. This proves the theorem. □

B.3.3 Claude

Proof. We prove this lower bound using a counting argument based on the fooling set method from communication complexity.

Let n and k be positive integers with $k \leq n$. Define $m = \lfloor n/k \rfloor$. We construct a fooling set of size m for the set disjointness problem.

Partition the set $[n]$ into m disjoint subsets $S_1, S_2, \ldots, S_m$ where each $|S_i| = k$ for $i = 1, 2, \ldots, m-1$, and $|S_m| \leq k$. This is possible since $mk \leq n < (m+1)k$.

For $i = 1, 2, \ldots, m$, define the input pair (A_i, B_i) where $A_i = S_i$ and $B_i = S_i$. Note that $A_i \cap B_i = S_i \neq \emptyset$ for all i.

We claim that $\mathcal{F} = \{(A_1, B_1), (A_2, B_2), \ldots, (A_m, B_m)\}$ forms a fooling set. To verify this, we must show that for any $i \neq j$, at least one of the pairs (A_i, B_j) or (A_j, B_i) has disjoint sets.

For $i \neq j$, we have $A_i = S_i$ and $B_j = S_j$. Since the sets $S_1, S_2, \ldots, S_m$ are pairwise disjoint by construction, we have $A_i \cap B_j = S_i \cap S_j = \emptyset$.

By the fooling set argument, any deterministic communication protocol that correctly determines whether two k-subsets of $[n]$ intersect must produce distinct transcripts for each pair in $\mathcal{F}$. This is because:

(1) All pairs (A_i, B_i) in $\mathcal{F}$ have the same output (intersecting sets).

(2) For $i \neq j$, the pair (A_i, B_j) has disjoint sets, giving a different output than (A_i, B_i).

If the protocol produced the same transcript for (A_i, B_i) and (A_j, B_j) with $i \neq j$, then it would also produce this transcript for (A_i, B_j). But (A_i, B_i) has intersecting sets while (A_i, B_j) has disjoint sets, contradicting correctness.

Therefore, the protocol must produce at least $m = \lfloor n/k \rfloor$ distinct transcripts. Since each transcript is a binary string, the minimum number of bits required is at least $\log_2(m) = \log_2(\lfloor n/k \rfloor)$.

This completes the proof that at least $\log(\lfloor n/k \rfloor)$ bits of communication are required. □

B.3.4 Gemini

Proof. Let n and k be positive integers such that $k \leq n$. Let $m = \lfloor n/k \rfloor$. We construct a family of subsets of $[n]$ to reduce the Equality problem to the Intersection problem. Since $mk \leq n$, we can partition a subset of $[n]$ of size mk into m pairwise disjoint sets, denoted by $S_1, S_2, \ldots, S_m$, where $|S_i| = k$ for each $i \in \{1, \ldots, m\}$.

Consider a deterministic communication protocol Π that correctly computes the function $f(A, B)$, defined to be 1 if $A \cap B \neq \emptyset$ and 0 otherwise, for all subsets $A, B \subseteq [n]$ of size k. We restrict our attention to the inputs where Alice receives a set $A \in \{S_1, \ldots, S_m\}$ and Bob receives a set $B \in \{S_1, \ldots, S_m\}$.

We employ the fooling set method for deterministic communication complexity. Consider the set of input pairs $\mathcal{F} = \{(S_i, S_i) \mid 1 \leq i \leq m\}$. For any pair $(S_i, S_i) \in \mathcal{F}$, we have $S_i \cap S_i = S_i \neq \emptyset$ (since $k \geq 1$), so $f(S_i, S_i) = 1$.

Now, consider any two distinct pairs (S_i, S_i) and (S_j, S_j) from $\mathcal{F}$, where $i \neq j$. By the construction of our sets, S_i and S_j are disjoint. Therefore, the "mixed" input (S_i, S_j) satisfies $S_i \cap S_j = \emptyset$, implying $f(S_i, S_j) = 0$.

In any deterministic communication protocol, the set of inputs leading to the same transcript (and thus the same leaf in the protocol tree) forms a combinatorial rectangle $\mathcal{X} \times \mathcal{Y}$. If two distinct inputs (S_i, S_i) and (S_j, S_j) resulted in the same transcript, they would belong to the same monochromatic rectangle. By the definition of a rectangle, this implies that the input (S_i, S_j) must also belong to that rectangle and thus generate the same transcript. Consequently, the protocol would output the same value for (S_i, S_j) as it does for (S_i, S_i). However, $f(S_i, S_i) = 1$ while $f(S_i, S_j) = 0$. This contradiction implies that no two pairs in $\mathcal{F}$ can generate the same transcript.

Since there are $|\mathcal{F}| = m$ distinct pairs in the fooling set, the protocol must generate at least m distinct transcripts. The number of bits of communication corresponds to the depth of the protocol tree. To have at least m distinct leaves, the binary tree must have depth d such that $2^d \geq m$. Therefore, the communication cost $C(\Pi)$ satisfies:

$$C(\Pi) \geq \lceil \log_2 m \rceil \geq \log_2 \left(\left\lfloor \frac{n}{k} \right\rfloor \right).$$

This completes the proof. □

B.4 Problem 1.4

B.4.1 Problem Statement

Theorem B.4. *Suppose Alice receives a string $x \in \{0,1\}^n$ that has more 0s than 1s, and Bob receives a string $y \in \{0,1\}^n$ that has more 1s than 0s. If they communicate to find a coordinate i where $x_i \neq y_i$, then at least $2 \log n$ bits of communication are required.*

B.4.2 GPT

Proof. Let the strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$ be given such that x contains more 0s than 1s, and y contains more 1s than 0s. We aim to show that at least $2 \log n$ bits of communication are required for Alice and Bob to find a coordinate i where $x_i \neq y_i$.

Let us define the following quantities: - z_x is the number of 0s in x, i.e., $z_x = \#\{i : x_i = 0\}$, and similarly, let z_y be the number of 1s in y, i.e., $z_y = \#\{i : y_i = 1\}$. - Let $S = \{1, 2, \ldots, n\}$ be the set of coordinates of the strings.

Since x has more 0s than 1s, we have $z_x > n/2$. Similarly, since y has more 1s than 0s, we have $z_y > n/2$.

We seek a coordinate $i \in S$ such that $x_i \neq y_i$. Since x contains more 0s than 1s and y contains more 1s than 0s, there must be at least one coordinate where $x_i \neq y_i$. Thus, it is guaranteed that x and y differ in at least one coordinate.

Let us now analyze the communication required for Alice and Bob to find such a coordinate. Alice knows the string x, and Bob knows the string y. To determine the coordinate i where $x_i \neq y_i$, Alice and Bob must exchange information.

Consider a protocol where Alice sends a message to Bob. The goal is to minimize the amount of communication while still ensuring that Bob can identify a coordinate i such that $x_i \neq y_i$. The key observation here is that the information about which coordinate differs between x and y depends on the positions where x and y differ, and these positions must be communicated effectively.

Given that x contains more 0s than 1s and y contains more 1s than 0s, the positions where $x_i \neq y_i$ are not immediately obvious without significant communication. In fact, since x and y are both of length n and differ in at least one coordinate, the number of possible positions where $x_i \neq y_i$ is at least $\frac{n}{2}$. Therefore, to find a specific i where $x_i \neq y_i$, Bob needs to receive enough information to distinguish between at least $\frac{n}{2}$ possible positions.

The amount of information required to distinguish between $\frac{n}{2}$ positions is at least $\log_2 \left(\frac{n}{2}\right)$ bits, which simplifies to:

$$\log_2 \left(\frac{n}{2}\right) = \log_2 n - 1.$$

Thus, the communication required for Bob to identify a coordinate i where $x_i \neq y_i$ is at least $\log_2 n - 1$ bits. Since Alice and Bob are performing a two-way communication, the total communication complexity is at least $2(\log_2 n - 1)$ bits.

Therefore, the total communication required is at least:

$$2(\log_2 n - 1) = 2\log_2 n - 2.$$

Hence, at least $2\log n$ bits of communication are required to find a coordinate i where $x_i \neq y_i$. This completes the proof. □

B.4.3 Claude

Proof. We prove this theorem using a fooling set argument from communication complexity theory.

Let $S_0 = \{x \in \{0,1\}^n : |x|_0 > |x|_1\}$ denote the set of strings with more 0s than 1s, and let $S_1 = \{y \in \{0,1\}^n : |y|_1 > |y|_0\}$ denote the set of strings with more 1s than 0s. Note that for any $x \in S_0$ and $y \in S_1$, we have $x \neq y$, which implies there exists at least one coordinate i where $x_i \neq y_i$.

We construct a fooling set $\mathcal{F} \subseteq S_0 \times S_1$ as follows. For each subset $T \subseteq [n]$ with $|T| = \lceil n/2 \rceil$, define $x^T \in \{0,1\}^n$ by setting $(x^T)_i = 0$ if $i \in T$ and $(x^T)_i = 1$ if $i \notin T$. Similarly, define $y^T \in \{0,1\}^n$ by setting $(y^T)_i = 1$ if $i \in T$ and $(y^T)_i = 0$ if $i \notin T$. Note that x^T has exactly $\lceil n/2 \rceil$ zeros and $\lfloor n/2 \rfloor$ ones, so $x^T \in S_0$. Similarly, y^T has exactly $\lceil n/2 \rceil$ ones and $\lfloor n/2 \rfloor$ zeros, so $y^T \in S_1$.

Let $\mathcal{F} = \{(x^T, y^T) : T \subseteq [n], |T| = \lceil n/2 \rceil\}$. The size of $\mathcal{F}$ is $|\mathcal{F}| = \binom{n}{\lceil n/2 \rceil}$.

We verify that $\mathcal{F}$ is a fooling set. Consider two distinct pairs (x^T, y^T) and $(x^{T'}, y^{T'})$ in $\mathcal{F}$ where $T \neq T'$. We claim that either x^T and $y^{T'}$ agree on all coordinates, or $x^{T'}$ and y^T agree on all coordinates.

Since $T \neq T'$, there exists $j \in T \triangle T'$ (the symmetric difference). Without loss of generality, assume $j \in T \setminus T'$. Then $(x^T)_j = 0$ and $(y^T)_j = 1$, while $(x^{T'})_j = 1$ and $(y^{T'})_j = 0$.

For any coordinate $i \in T \cap T'$, we have $(x^T)_i = 0 = (x^{T'})_i$ and $(y^T)_i = 1 = (y^{T'})_i$, so $(x^T)_i \neq (y^{T'})_i$ and $(x^{T'})_i \neq (y^T)_i$.

For any coordinate $i \in [n] \setminus (T \cup T')$, we have $(x^T)_i = 1 = (x^{T'})_i$ and $(y^T)_i = 0 = (y^{T'})_i$, so $(x^T)_i \neq (y^{T'})_i$ and $(x^{T'})_i \neq (y^T)_i$.

For $i \in T \setminus T'$, we have $(x^T)_i = 0$, $(y^{T'})_i = 0$, $(x^{T'})_i = 1$, and $(y^T)_i = 1$. Thus $(x^T)_i = (y^{T'})_i$ but $(x^{T'})_i \neq (y^T)_i$.

For $i \in T' \setminus T$, we have $(x^{T'})_i = 0$, $(y^T)_i = 0$, $(x^T)_i = 1$, and $(y^{T'})_i = 1$. Thus $(x^{T'})_i = (y^T)_i$ but $(x^T)_i \neq (y^{T'})_i$.

This shows that the pairs $(x^T, y^{T'})$ and $(x^{T'}, y^T)$ have different outputs, establishing that $\mathcal{F}$ is a fooling set.

By the fooling set lower bound in communication complexity, any deterministic protocol requires at least $\log_2 |\mathcal{F}| = \log_2 \binom{n}{\lceil n/2 \rceil}$ bits of communication.

Using Stirling's approximation, we have $\binom{n}{\lceil n/2 \rceil} \geq \frac{2^n}{\sqrt{n\pi}}$ for sufficiently large n. Therefore,

$$\log_2 \binom{n}{\lceil n/2 \rceil} \geq \log_2 \left(\frac{2^n}{\sqrt{n\pi}} \right) = n - \frac{1}{2}\log_2(n\pi) \geq n - \log_2 n - 2$$

for sufficiently large n. In particular, for $n \geq 4$, this is at least $2\log_2 n$.

Thus, any communication protocol solving this problem requires at least $2\log n$ bits of communication. □

B.4.4 Gemini

Proof. Let $f : \{0,1\}^n \to \{0,1\}$ be the Majority function, defined by $f(z) = 1$ if $\sum_{i=1}^n z_i > n/2$ and $f(z) = 0$ otherwise. The problem describes the Karchmer-Wigderson game associated with f. Let R_f denote the relation $\{(x, y, i) \mid f(x) = 0, f(y) = 1, x_i \neq y_i\}$. The Karchmer-Wigderson theorem states that the deterministic communication complexity of R_f, denoted $D(R_f)$, satisfies $D(R_f) = d(f)$, where $d(f)$ is the minimum depth of a boolean circuit (over the basis $\{\wedge, \vee, \neg\}$) computing f.

It is a standard result in circuit complexity that the depth of a function is related to its formula size $L(f)$ by the inequality $d(f) \geq \log_2 L(f)$. Therefore, to lower bound the communication complexity, it suffices to lower bound the formula size of the Majority function. We utilize Khrapchenko's Theorem,

which states that for any subsets $A \subseteq f^{-1}(0)$ and $B \subseteq f^{-1}(1)$,

$$L(f) \geq \frac{|S|^2}{|A| \cdot |B|},$$

where $S = \{(x, y) \in A \times B \mid \Delta(x, y) = 1\}$ is the set of pairs in $A \times B$ with Hamming distance 1.

Assume without loss of generality that n is odd. Let $k = \lfloor n/2 \rfloor = (n-1)/2$. We define the sets A and B as the layers of the hypercube adjacent to the majority boundary:

$$A = \left\{ x \in \{0,1\}^n \;\middle|\; \sum_{i=1}^{n} x_i = k \right\}, \quad B = \left\{ y \in \{0,1\}^n \;\middle|\; \sum_{i=1}^{n} y_i = k+1 \right\}.$$

By definition, for any $x \in A$, the number of 1s is $k < n/2$, so $f(x) = 0$. For any $y \in B$, the number of 1s is $k + 1 > n/2$, so $f(y) = 1$. Thus, $A \subseteq f^{-1}(0)$ and $B \subseteq f^{-1}(1)$.

The sizes of these sets are:

$$|A| = \binom{n}{k}, \quad |B| = \binom{n}{k+1}.$$

Since $n = 2k + 1$, we have $\binom{n}{k} = \binom{n}{n-k} = \binom{n}{k+1}$, so $|A| = |B|$.

Next, we determine the size of S. A pair (x, y) has Hamming distance 1 if and only if y is obtained from x by flipping exactly one bit. Since x has weight k and y has weight $k + 1$, this bit flip must change a 0 in x to a 1. An input $x \in A$ has $n - k$ zeros. Therefore, each $x \in A$ is connected to exactly $n - k$ distinct elements in B. The total number of edges is:

$$|S| = |A| \cdot (n - k).$$

Applying Khrapchenko's bound:

$$L(f) \geq \frac{(|A|(n-k))^2}{|A| \cdot |B|} = \frac{|A|^2 (n-k)^2}{|A|^2} = (n-k)^2.$$

Substituting $k = (n-1)/2$:

$$n - k = n - \frac{n-1}{2} = \frac{2n - n + 1}{2} = \frac{n+1}{2}.$$

Thus,

$$L(f) \geq \left(\frac{n+1}{2} \right)^2 \approx \frac{n^2}{4}.$$

Finally, returning to the communication complexity:

$$D(R_f) = d(f) \geq \log_2 L(f) \geq \log_2 \left(\frac{n^2}{4} \right) = \log_2(n^2) - \log_2(4) = 2 \log_2 n - 2.$$

Asymptotically, the communication complexity is at least $2 \log n$. □

B.5 Problem 1.5

B.5.1 Problem Statement

Theorem B.5. *The chromatic number $\chi(G)$ of a graph G is the minimum number of colors needed to color the vertices of G so that no two adjacent vertices have the same color.*

1. Consider the following communication problem. Alice gets a vertex x in G, Bob gets a vertex y in G, and they need to decide whether $\{x, y\}$ is an edge in G. Show that $\log \chi(G)$ is at most the deterministic communication complexity of this problem. Hint: Recall that (x, x) is never an edge of the graph.

2. Let H denote the graph on the same vertex set as G, such that $\{x, y\}$ is an edge of H if and only if there is a path of length 2 of the form x, z, y in G. Consider the following communication problem. Alice gets x, Bob gets y, and they are promised that $\{x, y\}$ is an edge of G. Their goal is for Alice to learn y and Bob to learn x. Give a protocol with $2 \log \chi(H)$ bits for accomplishing this.

B.5.2 GPT

Proof. We will prove both parts of the theorem in the following manner:

Part 1: Upper Bound on Deterministic Communication Complexity in Terms of Chromatic Number

Consider the communication problem where Alice gets a vertex $x \in V(G)$, and Bob gets a vertex $y \in V(G)$. The task is to decide whether $\{x, y\}$ is an edge in the graph G. We aim to show that the deterministic communication complexity of this problem is at most $\log \chi(G)$.

We begin by noting that a proper vertex coloring of G with $\chi(G)$ colors ensures that adjacent vertices receive different colors. Let the color of vertex x be denoted by $c(x)$ and the color of vertex y by $c(y)$. The key observation here is that if $\{x, y\}$ is an edge, then $c(x) \neq c(y)$, as adjacent vertices must be assigned different colors. Conversely, if $c(x) = c(y)$, then $\{x, y\}$ is not an edge, as the coloring is invalid for adjacent vertices.

Thus, knowing the color of vertex x (i.e., $c(x)$) and the color of vertex y (i.e., $c(y)$) enables Alice and Bob to determine whether $\{x, y\}$ is an edge in G. Since there are $\chi(G)$ distinct colors, it follows that $c(x)$ and $c(y)$ can each be represented using $\lceil \log_2 \chi(G) \rceil$ bits.

Now, consider the deterministic communication protocol where Alice sends her color $c(x)$ to Bob and Bob sends his color $c(y)$ to Alice. Once they have exchanged this information, they can simply compare the colors to decide whether $\{x, y\}$ is an edge. The total number of bits exchanged is $2\lceil \log_2 \chi(G) \rceil$ bits. Therefore, the deterministic communication complexity of the problem is at most $2\lceil \log_2 \chi(G) \rceil$ bits, which is asymptotically at most $O(\log \chi(G))$. This completes the proof of part 1.

Part 2: Communication Protocol for Alice and Bob with Graph H

Next, we address the second communication problem. We are given a graph H defined on the same vertex set as G, where $\{x, y\}$ is an edge in H if and only if there exists a path of length 2 of the form x, z, y in G for some $z \in V(G)$. Alice is given vertex x, Bob is given vertex y, and they are promised that $\{x, y\}$ is an edge in G, meaning there exists some vertex z such that $\{x, z\}$ and $\{z, y\}$ are both edges in G. Their goal is for Alice to learn y and for Bob to learn x.

We are asked to find a communication protocol that accomplishes this with $2 \log \chi(H)$ bits.

Since $\{x, y\}$ is an edge in H, there exists a vertex $z \in V(G)$ such that x and y are connected by a 2-path in G, i.e., $\{x, z\}$ and $\{z, y\}$ are edges in G. Therefore, Alice knows that there is some z such that x is adjacent to z and Bob knows that there is some z such that z is adjacent to y. The idea of the protocol is to use the coloring information of G to determine the vertices involved in the 2-path.

First, observe that the chromatic number $\chi(H)$ of the graph H is at most the chromatic number of G, since every edge in H corresponds to a 2-path in G and the vertex coloring of G naturally extends to a valid coloring of H (since paths of length 2 in G do not introduce any new conflicts for coloring in H). Therefore, $\chi(H) \leq \chi(G)$.

To construct the protocol, we first note that the coloring of G provides a partition of the vertices of G into $\chi(G)$ distinct color classes. Since Alice knows x and Bob knows y, they can use the coloring of G to communicate about the color class of z without directly revealing z. Specifically, Alice can send Bob a message that encodes the color class of the vertex z that satisfies the condition $\{x, z\}$ in G, and Bob can send Alice a message encoding the color class of the vertex z that satisfies the condition $\{z, y\}$ in G.

Since there are at most $\chi(H)$ distinct color classes, Alice and Bob each need $\log_2 \chi(H)$ bits to communicate the color class information. Therefore, the total number of bits exchanged in the communication protocol is $2 \log_2 \chi(H)$ bits.

Thus, we have provided a communication protocol for Alice to learn y and Bob to learn x using $2 \log \chi(H)$ bits. This completes the proof of part 2.

□

B.5.3 Claude

Proof. We prove both parts of the theorem.

Part 1: We show that $\log \chi(G) \leq D(f)$, where $D(f)$ denotes the deterministic communication complexity of deciding whether $\{x, y\}$ is an edge in G.

Let $c : V(G) \to \{1, 2, \ldots, \chi(G)\}$ be an optimal proper coloring of G. We construct a deterministic protocol as follows: Alice sends $c(x)$ to Bob using $\lceil \log \chi(G) \rceil$ bits. Bob then compares $c(x)$ with $c(y)$.

If $c(x) = c(y)$, then Bob outputs that $\{x, y\}$ is not an edge. This is correct because if $\{x, y\}$ were an edge, then by the definition of proper coloring, we would have $c(x) \neq c(y)$, which is a contradiction.

If $c(x) \neq c(y)$, Bob needs to determine whether $\{x, y\}$ is an edge. However, we use the hint: note that (x, x) is never an edge. If Alice receives x and Bob receives y with $x = y$, then $c(x) = c(y)$ and the pair is not an edge. Conversely, if $c(x) \neq c(y)$, then $x \neq y$, but this alone does not determine edge membership.

We refine the protocol: Alice sends $c(x)$ using $\lceil \log \chi(G) \rceil$ bits. If $c(x) = c(y)$, then Bob knows $\{x, y\}$ is not an edge (since either $x = y$ or they have the same color in a proper coloring). If $c(x) \neq c(y)$, the vertices could be adjacent or non-adjacent. However, the communication complexity $D(f)$ must be at least $\lceil \log \chi(G) \rceil$ because any protocol must distinguish between at least $\chi(G)$ different color classes. Therefore, $\log \chi(G) \leq D(f)$.

Part 2: We construct a protocol using $2 \log \chi(H)$ bits for Alice to learn y and Bob to learn x, given that $\{x, y\}$ is an edge in G.

Let $c_H : V(H) \to \{1, 2, \ldots, \chi(H)\}$ be an optimal proper coloring of H. The protocol proceeds as follows:

Alice sends $c_H(x)$ to Bob using $\lceil \log \chi(H) \rceil$ bits.

Bob sends $c_H(y)$ to Alice using $\lceil \log \chi(H) \rceil$ bits.

The total communication is $2\lceil \log \chi(H) \rceil \leq 2 \log \chi(H) + 2$ bits, which is $O(\log \chi(H))$ and satisfies the bound $2 \log \chi(H)$ asymptotically.

We now verify correctness. Since $\{x, y\}$ is an edge in G, we know that x and y are not connected by a path of length 2 in G through themselves. By definition of H, $\{x, y\}$ is an edge in H if and only if there exists $z \in V(G)$ such that both $\{x, z\}$ and $\{z, y\}$ are edges in G.

The key observation is that if $\{x, y\}$ is an edge in G, then $\{x, y\}$ is not an edge in H (since a path of length 2 from x to y would require an intermediate vertex $z \neq x, y$, but we are given they are adjacent in G). Therefore, $c_H(x) \neq c_H(y)$ is not guaranteed.

Actually, we reconsider: Given that $\{x, y\}$ is an edge in G and knowing $c_H(x)$ and $c_H(y)$, Alice can determine y from the set of Bob's neighbors in G that have color $c_H(y)$ in H, and similarly for Bob. Since the coloring c_H partitions vertices into independent sets in H, and the communication of colors allows both parties to narrow down the possibilities, the protocol succeeds with $2 \log \chi(H)$ bits of communication. □

B.5.4 Gemini

Proof. We prove the two parts of the theorem sequentially.

Part 1. Let P be a deterministic communication protocol that solves the adjacency problem for $G = (V, E)$. Let C denote the communication complexity of P. The transcript of the protocol partitions the input space $V \times V$ into a set of disjoint combinatorial rectangles $\mathcal{R} = \{R_1, \ldots, R_k\}$, where $k \leq 2^C$. For each rectangle $R_i = A_i \times B_i$ (where $A_i, B_i \subseteq V$), the protocol outputs a constant value $v_i \in \{0, 1\}$. Since the protocol is correct, $v_i = 1$ if and only if $\{x, y\} \in E$ for all $(x, y) \in R_i$.

Consider the set of diagonal inputs $D = \{(x, x) : x \in V\}$. Since the graph G contains no self-loops, $\{x, x\} \notin E$ for any $x \in V$. Consequently, for any rectangle R_i that has a non-empty intersection with D, the associated output must be $v_i = 0$.

We construct a vertex coloring of G based on this partition. For each vertex $x \in V$, the input (x, x) belongs to exactly one rectangle in $\mathcal{R}$. Let $c(x) = i$ if $(x, x) \in R_i$. We claim that c is a proper coloring of G. Suppose distinct vertices x and y are assigned the same color, i.e., $c(x) = c(y) = i$. This implies that $(x, x) \in R_i$ and $(y, y) \in R_i$. By the definition of a combinatorial rectangle, since $x \in A_i$ and $y \in B_i$, it follows that $(x, y) \in A_i \times B_i = R_i$. Since R_i contains diagonal elements, its output is $v_i = 0$. Therefore, the protocol declares that $\{x, y\}$ is not an edge. By the correctness of the protocol, $\{x, y\} \notin E$.

Thus, any set of vertices assigned the same color forms an independent set in G. The number of colors used is at most the number of rectangles covering the diagonal, which is bounded by the total number of rectangles k. Therefore, $\chi(G) \leq k \leq 2^C$. Taking the logarithm of both sides, we obtain $\log \chi(G) \leq C$.

Part 2. Let $c_H : V \to \{1, \ldots, \chi(H)\}$ be a proper vertex coloring of the graph H. We define a protocol where Alice and Bob exchange the colors of their respective vertices in the graph H. Specifically, Alice sends $c_H(x)$ to Bob, and Bob sends $c_H(y)$ to Alice. The number of bits required to encode a color is $\lceil \log \chi(H) \rceil$, so the total communication cost is $2\lceil \log \chi(H) \rceil$.

To prove correctness, we must show that Alice can uniquely identify y given x, the promise $\{x, y\} \in E(G)$, and the color $c_H(y)$. Let $N_G(x) = \{v \in V : \{x, v\} \in E(G)\}$ be the neighborhood of x in G. Alice knows that $y \in N_G(x)$. Suppose there exist two distinct vertices $z_1, z_2 \in N_G(x)$ such that $c_H(z_1) = c_H(z_2)$. Since z_1 and z_2 are both adjacent to x in G, the sequence z_1, x, z_2 constitutes a path of length 2 in G. By the definition of the graph H, the existence of this path implies that $\{z_1, z_2\}$ is an edge in H. Since c_H is a proper coloring of H, adjacent vertices must have distinct colors, which implies $c_H(z_1) \neq c_H(z_2)$. This contradicts the assumption.

Therefore, all vertices in $N_G(x)$ must have distinct colors under c_H. Upon receiving $c_H(y)$, Alice inspects $N_G(x)$ and selects the unique neighbor v such that $c_H(v) = c_H(y)$. This vertex must be y. By symmetry, Bob can uniquely identify x among the neighbors of y using the received color $c_H(x)$. □

B.6 Problem 1.6

B.6.1 Problem Statement

Theorem B.6. *Let Alice be given a set $X \subseteq [n]$ and Bob be given a set $Y \subseteq [n]$. Their goal is to compute the median of the set $X \cup Y$. The communication complexity of this problem is at least $\Omega(n/\log n)$.*

B.6.2 GPT

Proof. Let $X \subseteq [n]$ and $Y \subseteq [n]$ be two sets given to Alice and Bob, respectively. Their goal is to compute the median of the set $X \cup Y$. Since X and Y are subsets of $[n]$, we know that $X \cup Y$ is also a subset of $[n]$, and its size can vary between $|X| + |Y|$ and n.

We are interested in determining the communication complexity of this problem, i.e., the minimum amount of information that Alice and Bob must exchange in order to compute the median of $X \cup Y$.

Step 1: Information-Theoretic Lower Bound

We begin by noting that the number of possible sets $X \cup Y$ is at most 2^n. In order to compute the median of $X \cup Y$, Alice and Bob must be able to distinguish between these 2^n possibilities. Therefore, the communication complexity must be at least as large as the number of bits required to distinguish between these 2^n sets, which is $\Omega(\log 2^n) = \Omega(n)$ bits.

Step 2: Tight Bound for Communication Complexity

To establish a more precise bound, we employ a reduction argument. Specifically, we consider a related problem: determining the median of two sets A and B of size approximately $n/2$, where A and B are subsets of $[n]$. The communication complexity of this problem is well-known to be $\Omega(n/\log n)$.

Given that $X \cup Y$ is a union of two sets X and Y of size at most n, we can observe that any protocol for computing the median of $X \cup Y$ must be able to distinguish between $X \cup Y$ and any other set of size $|X| + |Y|$. Since $|X| + |Y|$ is at most n, and the set $X \cup Y$ may contain elements from different parts of $[n]$, the protocol must communicate information about the relative positions of elements in order to compute the median correctly.

Step 3: Conclusion

Thus, based on the information-theoretic lower bound and the related problem, we conclude that the communication complexity of computing the median of $X \cup Y$ is at least $\Omega\left(\frac{n}{\log n}\right)$. This completes the proof.

□

B.6.3 Claude

Proof. We prove the lower bound by a reduction from the Index problem. In the Index problem, Alice receives a string $a \in \{0, 1\}^m$ and Bob receives an index $i \in [m]$, and their goal is for Bob to output a_i. It is well-known that the one-way communication complexity of Index is $\Omega(m)$.

Let $m = n/(2 \log n)$. We construct a reduction as follows. Given an instance (a, i) of Index where $a \in \{0, 1\}^m$ and $i \in [m]$, we construct sets $X, Y \subseteq [n]$ for the median problem.

Define X based on Alice's string a. Partition $[n]$ into m consecutive blocks $B_1, B_2, \ldots, B_m$ where each block B_j contains $2\log n$ consecutive integers. Specifically, $B_j = \{(j-1)\cdot 2\log n + 1, (j-1)\cdot 2\log n + 2, \ldots, j \cdot 2\log n\}$.

For each $j \in [m]$, Alice includes in X exactly $\log n$ elements from block B_j as follows:

- If $a_j = 0$, include the first $\log n$ elements of B_j in X.
- If $a_j = 1$, include the last $\log n$ elements of B_j in X.

Thus $|X| = m \log n = n/2$.

Bob constructs Y based on index i. For each block B_j with $j \neq i$, Bob includes in Y the $\log n$ elements from B_j that are not in X (Bob can determine this by choosing arbitrarily since these blocks do not affect the answer). For block B_i, Bob includes the first $\log n - 1$ elements and the last $\log n - 1$ elements of B_i in Y, giving $2\log n - 2$ elements from B_i.

Thus $|Y| = (m-1)\log n + (2\log n - 2) = m \log n + \log n - 2 = n/2 + \log n - 2$.

The set $X \cup Y$ contains all elements from blocks B_j for $j \neq i$, and from block B_i it contains:

- If $a_i = 0$: the first $\log n$ elements (from X) and the first $\log n - 1$ plus last $\log n - 1$ elements (from Y), giving all but the middle two elements of B_i.
- If $a_i = 1$: the last $\log n$ elements (from X) and the first $\log n - 1$ plus last $\log n - 1$ elements (from Y), giving all but the middle two elements of B_i.

The size of $X \cup Y$ is $(m-1)\cdot 2\log n + (2\log n - 2) = 2m\log n - 2 = n - 2$.

The median of $X \cup Y$ (the $\lceil (n-2)/2 \rceil$-th smallest element) falls within block B_i and its exact position depends on a_i. Specifically, the median distinguishes whether the missing elements are in the first half or second half of B_i, which reveals a_i.

Since any protocol for computing the median of $X \cup Y$ allows Bob to determine a_i, and the Index problem requires $\Omega(m) = \Omega(n/\log n)$ bits of communication, the median problem requires at least $\Omega(n/\log n)$ bits of communication. □

B.6.4 Gemini

Proof. We prove the lower bound by a reduction from the INDEX problem in the one-way communication model. In the INDEX problem, Alice holds a string $z \in \{0,1\}^m$ and Bob holds an index $k \in [m]$. Bob wishes to determine z_k. It is a standard result that the one-way randomized communication complexity of INDEX is $\Omega(m)$.

Let $m = \lfloor n/4 \rfloor$. Given an instance of the INDEX problem, Alice and Bob construct inputs $X \subseteq [n]$ and $Y \subseteq [n]$ for the Median problem as follows:

1. Alice constructs the set X based on her string z. She maps each bit z_i to an integer value in the range $[n/2+1, n]$. Specifically, for each $i \in [m]$, she includes the element $x_i = n/2 + 2i + z_i$ in X. Thus,

$$X = \{n/2 + 2i + z_i \mid i \in \{1, \ldots, m\}\}.$$

Note that since $1 \leq i \leq m \leq n/4$, the elements are distinct and contained in $[n/2+2, n]$. The size of X is exactly m. The elements of X preserve the order of indices i, i.e., $x_1 < x_2 < \cdots < x_m$.

2. Bob, wanting to determine z_k, constructs a set $Y \subseteq [n]$ to force the median of $X \cup Y$ to correspond to the element $x_k \in X$. Bob chooses Y to consist of L elements strictly smaller than any element in X. Specifically, Bob chooses $Y = \{1, 2, \ldots, L\} \subseteq [n/2]$.

Bob selects L such that the median of the set $S = X \cup Y$ is exactly the k-th element of X (which is x_k). The total size of the set S is $|S| = m + L$. The median of S is the element with rank $r = \lceil (m+L+1)/2 \rceil$ in the sorted order of S.

Since all elements of Y are smaller than all elements of X, the first L elements of S are the elements of Y, and the subsequent elements are $x_1, \ldots, x_m$. Therefore, the element with rank r in S is the element with rank $r - L$ in X. Bob sets L such that $r - L = k$.

Substituting r:

$$\left\lceil \frac{m+L+1}{2} \right\rceil - L = k \implies \frac{m+L+1}{2} \approx k + L \implies m + 1 - 2k \approx L.$$

Bob can choose L precisely to satisfy the rank condition (possibly adjusting by 1 depending on parity, which is always possible as Bob knows m and k). Since $k \in [m]$, L will be roughly $m - 2k$. To ensure $L \geq 0$, we can pad the original problem or assume Alice's indices are shifted, but asymptotically this holds for a constant fraction of indices. Alternatively, we can map the problem to a universe of size $O(n)$ and ensure sufficient padding space exists.

3. Upon computing the median of $X \cup Y$, the protocol outputs $x_k = n/2 + 2k + z_k$. Bob knows n and k, so he can subtract $n/2 + 2k$ from the median value to retrieve z_k.

Since the Median protocol allows Bob to solve the INDEX problem, the communication complexity of computing the median is lower-bounded by the complexity of INDEX, which is $\Omega(m) = \Omega(n)$. Since $\Omega(n) \implies \Omega(n/\log n)$, the theorem holds. □

B.7 Problem 1.7

B.7.1 Problem Statement

Theorem B.7. *Show that for every $0 \leq \alpha < 1/2$, any deterministic protocol for estimating the Hamming distance between two strings $x, y \in \{0,1\}^n$ up to an additive error αn must have length at least $\Omega(n)$. To do this, use an error correcting code. An error correcting code is a map $E : \{0,1\}^m \to \{0,1\}^{O(m)}$ with the property that if $x \neq y$, then $E(x)$ and $E(y)$ disagree in at least α fraction of their coordinates. Such a code exists for every $0 \leq \alpha < 1/2$. Use the code to prove the lower bound.*

B.7.2 GPT

Proof. Let $x, y \in \{0,1\}^n$ be two strings, and let $d(x,y)$ denote the Hamming distance between x and y. We are given that we wish to estimate $d(x,y)$ up to an additive error of αn for $0 \leq \alpha < 1/2$. Let us assume, for the sake of contradiction, that there exists a deterministic protocol for estimating $d(x,y)$ up to an additive error of αn that uses fewer than n bits.

We will now construct an error-correcting code and use it to prove that such a protocol must have length at least $\Omega(n)$. Let $E : \{0,1\}^m \to \{0,1\}^{O(m)}$ be an error-correcting code with the property that for any two distinct messages $x, y \in \{0,1\}^m$, the corresponding codewords $E(x)$ and $E(y)$ differ in at least an α fraction of their coordinates. Such a code exists for every $0 \leq \alpha < 1/2$ due to the existence of error-correcting codes with good distance properties.

Next, suppose we encode the two strings x and y using this error-correcting code. The encoded messages are $E(x)$ and $E(y)$, which are of length $O(m)$ by the construction of the code. Since $E(x)$ and $E(y)$ differ in at least an α fraction of their coordinates, it follows that the Hamming distance between $E(x)$ and $E(y)$ is at least $\alpha O(m)$.

Now, suppose we have a protocol that estimates the Hamming distance $d(x,y)$ between x and y within an additive error of αn. Let the protocol operate on the strings x and y, and suppose the protocol outputs a value $\widehat{d}(x,y)$, which is an estimate of the true Hamming distance. The protocol has a limited length, say L, which is strictly less than n. This implies that there are at most 2^L possible outputs of the protocol.

If the protocol fails to distinguish between different pairs of strings x, y in $\{0,1\}^n$ with sufficiently large Hamming distance, then we can use the protocol to distinguish between different codewords in the error-correcting code. Specifically, if the protocol can only output 2^L possible values, then there must exist distinct pairs of strings $x, y \in \{0,1\}^m$ such that their codewords $E(x)$ and $E(y)$ map to the same output in the protocol.

However, this leads to a contradiction. Recall that the codewords $E(x)$ and $E(y)$ differ in at least $\alpha O(m)$ coordinates. This implies that the Hamming distance between $E(x)$ and $E(y)$ is large, and thus the protocol should output distinct values for these two codewords. Since the protocol only has 2^L possible outputs, it cannot distinguish between all pairs of distinct strings in $\{0,1\}^n$, unless the protocol length L is large enough to accommodate this distinction.

Hence, for the protocol to distinguish between all pairs of distinct strings in $\{0,1\}^n$, the length of the protocol must satisfy $L \geq \Omega(n)$, as otherwise, it cannot distinguish between codewords that differ by a significant fraction of their coordinates.

Therefore, we conclude that any deterministic protocol for estimating the Hamming distance between two strings x and y up to an additive error of αn must have length at least $\Omega(n)$. □

B.7.3 Claude

Proof. We prove that any deterministic protocol for estimating the Hamming distance between two strings $x, y \in \{0,1\}^n$ up to an additive error αn must have communication complexity at least $\Omega(n)$ for any $0 \leq \alpha < 1/2$.

Let $E : \{0,1\}^m \to \{0,1\}^n$ be an error correcting code with the property that for any $x \neq y \in \{0,1\}^m$, the codewords $E(x)$ and $E(y)$ disagree in at least α fraction of their coordinates. Such a code exists for every $0 \leq \alpha < 1/2$ with $n = O(m)$.

Suppose, for the sake of contradiction, that there exists a deterministic protocol Π that estimates the Hamming distance between strings in $\{0,1\}^n$ up to additive error αn with communication complexity $c < m$ bits.

We construct a protocol for the equality problem on m-bit strings as follows. Given inputs $x, y \in \{0,1\}^m$, Alice computes $E(x)$ and Bob computes $E(y)$. They then run protocol Π on $E(x)$ and $E(y)$ to obtain an estimate $\widehat{d}$ of the Hamming distance $d_H(E(x), E(y))$.

By the guarantee of protocol Π, we have

$$|d_H(E(x), E(y)) - \widehat{d}| \leq \alpha n.$$

Now we analyze two cases:

Case 1: If $x = y$, then $E(x) = E(y)$, so $d_H(E(x), E(y)) = 0$. Therefore, $\widehat{d} \leq \alpha n$.

Case 2: If $x \neq y$, then by the error correcting property of the code, $d_H(E(x), E(y)) \geq \alpha n$. Since $\widehat{d}$ approximates $d_H(E(x), E(y))$ within additive error αn, we have

$$\widehat{d} \geq d_H(E(x), E(y)) - \alpha n \geq \alpha n - \alpha n = 0.$$

However, more importantly, we have

$$\widehat{d} \geq \alpha n - \alpha n = 0,$$

but since $d_H(E(x), E(y)) \geq \alpha n$, we must have $\widehat{d} \geq 0$.

To distinguish the cases properly, note that if $x \neq y$, then $d_H(E(x), E(y)) \geq \alpha n$, which implies $\widehat{d} \geq 0$. But we need a gap. Since the code has minimum distance at least αn, and the approximation error is at most αn, when $x \neq y$, we cannot guarantee $\widehat{d} > \alpha n$ in the worst case.

Let us reconsider with a better code. For $0 \leq \alpha < 1/2$, there exists an error correcting code $E : \{0,1\}^m \to \{0,1\}^n$ where $n = O(m)$ such that for $x \neq y$, we have $d_H(E(x), E(y)) \geq (1/2 - \alpha/2)n$.

With this code, if $x = y$, then $d_H(E(x), E(y)) = 0$, so $\widehat{d} \leq \alpha n$.

If $x \neq y$, then $d_H(E(x), E(y)) \geq (1/2 - \alpha/2)n$, so

$$\widehat{d} \geq (1/2 - \alpha/2)n - \alpha n = (1/2 - 3\alpha/2)n.$$

For $\alpha < 1/3$, we have $(1/2 - 3\alpha/2)n > \alpha n$, allowing us to distinguish the cases.

Thus, the protocol Π solves the equality problem on m-bit strings with communication c bits. By the well-known lower bound for the equality problem, we must have $c \geq \Omega(m)$. Since $n = O(m)$, this implies $c \geq \Omega(n)$.

Therefore, any deterministic protocol for estimating Hamming distance up to additive error αn must have communication complexity at least $\Omega(n)$. □

B.7.4 Gemini

Proof. We prove the lower bound by a reduction from the Equality function. Let $\text{EQ}_m : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ be the function defined by $\text{EQ}_m(u,v) = 1$ if $u = v$ and $\text{EQ}_m(u,v) = 0$ if $u \neq v$. It is a standard result in communication complexity that the deterministic communication complexity of the Equality function is lower bounded by m, i.e., $D(\text{EQ}_m) \geq m$.

Let Π be a deterministic protocol that estimates the Hamming distance between two strings $x, y \in \{0,1\}^n$ up to an additive error αn. That is, if $\Pi(x,y)$ denotes the output of the protocol, then

$$|\Pi(x,y) - d_H(x,y)| \leq \alpha n,$$

where $d_H(x,y)$ is the Hamming distance between x and y.

We construct a protocol for EQ_m using Π and an error correcting code. Let $E : \{0,1\}^m \to \{0,1\}^n$ be an error correcting code with block length $n = O(m)$ (implying $m = \Omega(n)$) and minimum relative distance δ. The property of the code is that for any distinct $u, v \in \{0,1\}^m$, the Hamming distance between the codewords satisfies

$$d_H(E(u), E(v)) \geq \delta n.$$

To ensure the reduction works, we select a code such that $\delta > 2\alpha$. Note that for any $\alpha < 1/4$, such a code exists (e.g., codes on the Gilbert-Varshamov bound exist with relative distance approaching $1/2$).

The protocol for $\text{EQ}_m(u,v)$ proceeds as follows:

1. Alice holds $u \in \{0,1\}^m$ and computes $x = E(u) \in \{0,1\}^n$.
2. Bob holds $v \in \{0,1\}^m$ and computes $y = E(v) \in \{0,1\}^n$.
3. Alice and Bob run the protocol Π on inputs x and y. Let $Z = \Pi(x,y)$ be the output.
4. If $Z \leq \alpha n$, they output 1 (indicating $u = v$). If $Z > \alpha n$, they output 0 (indicating $u \neq v$).

We analyze the correctness of this protocol:

- **Case 1:** $u = v$**.** In this case, $x = y$, so $d_H(x,y) = 0$. The estimator Π guarantees that $|Z - 0| \leq \alpha n$, which implies $Z \leq \alpha n$. The protocol correctly outputs 1.
- **Case 2:** $u \neq v$**.** In this case, $d_H(x,y) \geq \delta n$. The estimator Π guarantees that $|Z - d_H(x,y)| \leq \alpha n$, which implies $Z \geq d_H(x,y) - \alpha n \geq \delta n - \alpha n$. For the protocol to correctly output 0, we require $Z > \alpha n$. A sufficient condition is:

$$\delta n - \alpha n > \alpha n \iff \delta > 2\alpha.$$

Assuming a code is chosen with relative distance $\delta > 2\alpha$, the constructed protocol correctly computes $\text{EQ}_m(u,v)$ deterministically.

Since the communication cost of the constructed protocol is exactly the length of Π, denoted $L(\Pi)$, and we know $D(\text{EQ}_m) \geq m$, it follows that:

$$L(\Pi) \geq m.$$

Given that the code maps m bits to $n = O(m)$ bits, we have $m = \Omega(n)$. Therefore:

$$L(\Pi) = \Omega(n).$$

This completes the proof. □

B.8 Problem 1.8

B.8.1 Problem Statement

Theorem B.8. *Suppose Alice is given a string $x \in [w]^k$, and Bob is given a sequence Y of sets $Y_1, \ldots, Y_k \subseteq [w]$. If there is a protocol that determines whether or not there is an i such that $x_i \in Y_i$, with Alice sending a bits and Bob sending b bits, then*

$$a + b \geq \frac{wk}{2^{a/k} + 1}.$$

B.8.2 GPT

Proof. Let us consider the protocol in which Alice is given a string $x \in [w]^k$ and Bob is given a sequence $Y_1, \ldots, Y_k \subseteq [w]$. The goal of the protocol is to determine whether there exists an index $i \in \{1, \ldots, k\}$ such that $x_i \in Y_i$. The communication complexity of this protocol is given by Alice sending a bits and Bob sending b bits.

Suppose that there exists a protocol which solves this problem. The communication complexity of the protocol must satisfy the following lower bound, which we will derive.

The total number of possible inputs to Alice is the number of distinct strings of length k over an alphabet of size w, which is w^k. Similarly, the total number of possible inputs to Bob is the total number of sequences of subsets $Y_1, \ldots, Y_k$, where each $Y_i \subseteq [w]$. Since each subset Y_i can have at most w elements, there are at most 2^w choices for each Y_i. Thus, the total number of inputs for Bob is $(2^w)^k = 2^{wk}$.

The communication complexity of the protocol is related to the number of distinct inputs Alice and Bob can exchange. By the communication protocol's definition, the number of distinct messages that Alice can send is at most 2^a, and the number of distinct messages that Bob can send is at most 2^b. Therefore, the total number of distinct message pairs (i.e., the number of distinct communications between Alice and Bob) is at most $2^a \times 2^b$.

For the protocol to determine whether there exists an index i such that $x_i \in Y_i$, it must distinguish between all possible input pairs, i.e., between all w^k possible inputs for Alice and all 2^{wk} possible inputs for Bob. Therefore, the total number of distinct message pairs must be at least the number of distinct input pairs, giving the inequality:

$$2^a \times 2^b \geq w^k \times 2^{wk}.$$

Taking the logarithm of both sides, we obtain:

$$a + b \geq wk + wk.$$

Simplifying, we get:

$$a + b \geq 2wk.$$

Now, consider the case where Alice and Bob follow a strategy where they use a bits for Alice and b bits for Bob. Given the constraints on the inputs, it is known that the total communication complexity must satisfy the inequality:

$$a + b \geq \frac{wk}{2a/k + 1}.$$

Therefore, combining the two inequalities, we conclude that the communication complexity of the protocol is lower-bounded by:

$$a + b \geq \frac{wk}{2a/k + 1}.$$

This completes the proof. □

B.8.3 Claude

Proof. We prove this lower bound using an information-theoretic argument based on counting the number of possible inputs and outputs.

Let $\mathcal{P}$ denote the protocol where Alice sends a bits and Bob sends b bits. The protocol must correctly determine whether there exists an index $i \in [k]$ such that $x_i \in Y_i$.

First, we count the total number of possible inputs. Alice's input $x \in [w]^k$ can take w^k possible values. For each fixed Alice input x, we consider Bob's inputs $Y = (Y_1, \ldots, Y_k)$ where each $Y_i \subseteq [w]$.

For a fixed string $x \in [w]^k$, we partition Bob's possible inputs into two classes:

(i) Inputs Y such that $x_i \in Y_i$ for some $i \in [k]$ (the protocol outputs YES)

(ii) Inputs Y such that $x_i \notin Y_i$ for all $i \in [k]$ (the protocol outputs NO)

The number of Bob's inputs in class (ii) is exactly $(w-1)^k$, since for each coordinate i, the set Y_i must avoid the specific element x_i, giving 2^{w-1} choices per coordinate.

Now we apply an information-theoretic argument. Alice's message partitions her input space $[w]^k$ into at most 2^a cells. For the protocol to be correct, if two inputs x and x' of Alice lead to the same message, then for any Bob input Y, the protocol must give the same answer for both (x, Y) and (x', Y) based solely on Bob's computation after receiving Alice's message.

Consider a fixed message m from Alice, which corresponds to some subset $S_m \subseteq [w]^k$ of Alice's inputs. For each $x \in S_m$, there are $(w-1)^k$ Bob inputs Y for which the answer should be NO.

By a pigeonhole argument, if $|S_m|$ is large, then there must be significant overlap in the Bob inputs that should yield NO for different $x \in S_m$. However, Bob must distinguish between all possible scenarios using only b bits.

More precisely, for each message m from Alice with corresponding set S_m, Bob must be able to distinguish among all pairs (x, Y) where $x \in S_m$. The number of such pairs where the answer is NO is at least $|S_m| \cdot (w-1)^k$.

Since Bob sends b bits, he can produce at most 2^b different messages. For the protocol to be correct, Bob's message must allow both parties to determine the correct answer. This means that the 2^b possible outputs from Bob must be sufficient to encode the answer for all relevant input pairs.

By an averaging argument, there exists a message m from Alice such that $|S_m| \geq w^k/2^a$. For this message, the number of NO instances is at least $(w^k/2^a) \cdot (w-1)^k$.

For Bob to correctly handle all inputs after receiving message m, we need 2^b to be large enough to distinguish the necessary cases. The total number of distinct Bob inputs is 2^{wk}, but Bob must correctly classify them into YES/NO categories.

Using a more refined counting argument: the protocol induces a partition of the input space. The number of YES instances for a fixed x is $2^{wk} - (2^{w-1})^k$. By information theory, we need:

$$2^{a+b} \geq w^k \cdot (2^{w-1})^k / (2^{wk} - (2^{w-1})^k)$$

Through careful analysis of the information-theoretic constraints and using the fact that the protocol must distinguish $(w-1)^k$ NO instances for each Alice input, we obtain:

$$2^{a+b} \geq w^k \cdot \frac{(w-1)^k}{2^{wk}}$$

Simplifying and using the approximation $(w-1)/w \approx 1 - 1/w$ for large w, and applying logarithmic analysis, we derive:

$$a + b \geq k \log_2(w) - a$$

which gives us:

$$2a + b \geq k \log_2(w)$$

By optimizing over the communication pattern and using the constraint that Alice sends a bits partitioning her space into 2^a cells, each of average size $w^k/2^a$, and Bob must distinguish $(w-1)^k$ cases, we obtain:

$$a + b \geq \frac{wk}{2^{a/k} + 1}$$

□

B.8.4 Gemini

Proof. Let Π be the protocol where Alice sends a bits and Bob sends b bits. Alice's message $M_A(x)$ depends only on x, and Bob's message $M_B(Y)$ depends only on Y (without loss of generality, as Bob's message in a general protocol can be viewed as partitioning the space of Y for each fixed transcript prefix from Alice). The final output is a function of the transcript $(M_A(x), M_B(Y))$.

There are 2^a possible messages from Alice. The set of inputs $[w]^k$ is partitioned into sets $X_m = \{x \in [w]^k \mid M_A(x) = m\}$ for $m \in \{0,1\}^a$. By the Pigeonhole Principle, there exists a message m^* such that

$$|X_{m^*}| \geq \frac{w^k}{2^a}.$$

Let $X = X_{m^*}$. For each coordinate $i \in \{1, \ldots, k\}$, let $S_i = \{x_i \mid x \in X\} \subseteq [w]$ be the projection of X onto the i-th component. Let $s_i = |S_i|$. Since $X \subseteq S_1 \times S_2 \times \cdots \times S_k$, we have

$$\prod_{i=1}^{k} s_i \geq |X| \geq \frac{w^k}{2^a}.$$

Using the AM-GM inequality, the arithmetic mean of the sizes s_i satisfies:

$$\frac{1}{k}\sum_{i=1}^{k} s_i \geq \left(\prod_{i=1}^{k} s_i\right)^{1/k} \geq \left(\frac{w^k}{2^a}\right)^{1/k} = \frac{w}{2^{a/k}}.$$

Thus, $\sum_{i=1}^{k} s_i \geq kw2^{-a/k}$.

Now consider the behavior of Bob's messages on the set of inputs X. Since Alice's message is fixed to m^*, the protocol's output depends solely on Bob's message $M_B(Y)$. Let $r = M_B(Y)$. The protocol outputs a value $Out(m^*, r) \in \{0, 1\}$. For the protocol to be correct, for any fixed Y, the output must be correct for all $x \in X$. Specifically, consider the case where the protocol outputs 0 (indicating $\forall i, x_i \notin Y_i$). If Bob sends a message r such that $Out(m^*, r) = 0$, then for every Y that maps to r, it must be that for all $x \in X, \forall i, x_i \notin Y_i$. This is equivalent to saying that for all i, $Y_i \cap S_i = \emptyset$. Conversely, if Y is such that $Y_i \cap S_i \neq \emptyset$ for some i, Bob cannot send a message that maps to 0.

Let $U = \bigcup_{i=1}^{k}(\{i\} \times S_i) \subseteq \{1, \ldots, k\} \times [w]$. The size of U is $|U| = \sum_{i=1}^{k} s_i$. Consider the family of sets Y defined by subsets of U. For any subset $Z \subseteq U$, define a sequence Y^Z by $Y_i^Z = \{v \in [w] \mid (i, v) \in Z\}$. Note that $Y_i^Z \subseteq S_i$. For any such Y^Z, the intersection with X is determined by the structure of X. However, we consider the specific "singleton" sets. For any element $u = (j, v) \in U$, let $Y^{\{u\}}$ be the sequence where $Y_j = \{v\}$ and $Y_l = \emptyset$ for $l \neq j$. Since $v \in S_j$, there exists some $x \in X$ such that $x_j = v$. Thus, for input $(x, Y^{\{u\}})$, the intersection is non-empty (output 1). For $Y^{\emptyset} = (\emptyset, \ldots, \emptyset)$, the intersection is empty for all x (output 0).

Bob's message must distinguish the case $Y^{\emptyset}$ from any case where intersection is guaranteed. More robustly, consider the set of all possible sequences Y. Bob sends b bits, partitioning the space of Y into 2^b classes. Let $\mathcal{Y}_r = \{Y \mid M_B(Y) = r\}$. For any class $\mathcal{Y}_r$, the protocol predicts a constant output for all $x \in X$. If the predicted output is 0, then every $Y \in \mathcal{Y}_r$ must satisfy $Y_i \cap S_i = \emptyset$ for all i. If the predicted output is 1, then every $Y \in \mathcal{Y}_r$ must satisfy $\exists i, x_i \in Y_i$ for all $x \in X$.

We employ a covering argument. The total number of possible sequences Y is 2^{wk}. However, we focus on the information theoretic content required to handle the projections S_i. It is a known result in communication complexity for the "membership" or "indexing" problem that to handle a universe of size $S = \sum s_i$, if Alice has the element (here represented by the choice of $x \in X$ effectively selecting elements in S_i) and Bob has the set, Bob must send b bits such that $a + b$ covers the complexity. Specifically, for the bit-probe complexity or the asymmetric communication complexity of the vector indexing problem, the lower bound is derived from the fact that Bob must distinguish between Y's that intersect S_i and those that don't. Consider the set of Y's such that $Y_i \subseteq S_i$. There are $2^{\sum s_i}$ such sets. For any $Z \subseteq U$, let Y^Z be as defined above. If X were the full box $\prod S_i$, then for any non-empty Z, there exists $x \in X$ intersecting Y^Z. Thus Y^Z (answer 1) must be distinguished from $Y^{\emptyset}$ (answer 0). While X is not necessarily the full box, the lower bound $a + b \geq \frac{wk}{2^{a/k}+1}$ holds. Let us derive the inequality from the size constraint. We established $\sum s_i \geq kw2^{-a/k}$. Suppose for contradiction that $a + b < \frac{wk}{2^{a/k}+1}$. Let $t = 2^{a/k}$. Then $a = k \log t$. The condition is $k \log t + b < \frac{wk}{t+1}$. This implies $b < \frac{wk}{t+1} - k \log t$. However, we must have $b \geq \sum s_i - (wk - \frac{wk}{t+1})$? No. The standard lower bound for this problem states that $b \geq \sum s_i - k$. Substituting the lower bound for $\sum s_i$:

$$b \geq kw2^{-a/k} - k.$$

Let us check if $a + kw2^{-a/k} - k \geq \frac{wk}{2^{a/k}+1}$. Let $u = 2^{a/k}$. We examine $k \log u + \frac{kw}{u} - k$ vs $\frac{kw}{u+1}$. For large w, the linear term in w dominates. The term $\frac{kw}{u}$ is strictly greater than $\frac{kw}{u+1}$. Thus, for sufficiently large w, the bound holds. More precisely, the inequality $a + b \geq \frac{wk}{2^{a/k}+1}$ is a direct consequence of the trade-off between the size of the sub-universe $\sum s_i$ (which decreases with a) and Bob's communication b. Specifically, we have the bound $b \geq \sum_{i=1}^{k} s_i - (wk - \text{something})$. Actually, the tightest combinatorial bound for the "Or of Ands" (which this problem generalizes) implies that if Alice sends a bits, Bob must send b bits such that $(a + b)(2^{a/k} + 1) \geq wk$. Rearranging this gives $a + b \geq \frac{wk}{2^{a/k}+1}$.

To complete the formal proof steps: 1. Fix Alice's message m to maximize $|X_m|$, obtaining projections S_i with $\sum s_i \geq kw2^{-a/k}$. 2. Restrict the problem to inputs $x \in X_m$ and Y where $Y_i \subseteq S_i$. 3. This embeds a set intersection problem on a universe of size $M = \sum s_i$. 4. For the protocol to correctly determine intersection on this sub-universe, the total communication C (given m is fixed, $C = b$) must satisfy $b \geq M$ in the deterministic case for exact set identification, or more loosely for the intersection decision problem with one-sided error (which the 0-output case implies), b must be close to M. 5. Specifically, the bound $b \geq \sum s_i - k$ holds for distinguishing the empty set from singletons. 6. Therefore, $a + b \geq a + \sum s_i - k \geq a + kw2^{-a/k} - k$. 7. We verify that $a + kw2^{-a/k} \geq \frac{wk}{2^{a/k}+1}$. Let $f(a) = a + kw2^{-a/k}$. We want to show $f(a) \geq \frac{wk}{2^{a/k}+1}$. Since $2^{a/k} \geq 1$, $\frac{wk}{2^{a/k}+1} \leq \frac{wk}{2^{a/k}} = kw2^{-a/k}$. Thus, $a + b \geq kw2^{-a/k} \geq \frac{wk}{2^{a/k}+1}$ is true (ignoring the $-k$ term which is negligible for large w). For the exact non-asymptotic bound, the rigorous combinatorial lemma (Miltersen et al.) states that for any protocol:

$$b \geq \frac{wk - a(2^{a/k} + 1)}{2^{a/k} + 1} \implies b(2^{a/k} + 1) \geq wk - a(2^{a/k} + 1) \implies (a + b)(2^{a/k} + 1) \geq wk.$$

This yields the stated inequality. □

B.9 Problem 1.9

B.9.1 Problem Statement

Theorem B.9. *Given a function $f : \{0,1\}^{2n} \to \{0,1\}$ and a subset $S \subseteq [2n]$, let $C(f, S)$ be the communication complexity of computing f, when Alice is given the bits of the input that correspond to S, and Bob is given the bits that correspond to the complement of S.*

There exists a universal constant $\epsilon > 0$ such that for most functions f, it holds that

$$C(f, S) \geq \epsilon n$$

for every set S with $2n/3 \leq |S| \leq 3n/4$.

B.9.2 GPT

Proof. Let $f : \{0,1\}^{2n} \to \{0,1\}$ be a Boolean function, and let $S \subseteq [2n]$ be a subset of indices. We are tasked with proving that for most functions f, there exists a universal constant $\epsilon > 0$ such that the communication complexity $C(f, S)$ satisfies the inequality

$$C(f, S) \geq \epsilon n$$

for every set S with $2n/3 \leq |S| \leq 3n/4$. Here, $C(f, S)$ is defined as the communication complexity of computing f, where Alice is given the bits of the input corresponding to the indices in S, and Bob is given the bits of the input corresponding to the complement of S.

Step 1: High-Level Intuition

The communication complexity $C(f, S)$ measures the amount of communication required between Alice and Bob to compute the value of f given their respective parts of the input. Our goal is to show that for most Boolean functions f, for any subset S of the input bits with size between $2n/3$ and $3n/4$, the communication complexity must be at least some constant fraction of n.

Step 2: Random Functions and Communication Complexity

Consider a random Boolean function $f : \{0,1\}^{2n} \to \{0,1\}$, chosen uniformly at random. The key observation is that, for such random functions, the communication complexity $C(f, S)$ is closely related to the sensitivity of the function with respect to the partition of the input bits between Alice and Bob.

Let S be a subset of indices such that $2n/3 \leq |S| \leq 3n/4$. For these sets, we will show that with high probability, the communication complexity $C(f, S)$ is large. Specifically, for random functions, the communication complexity is expected to be at least a constant fraction of n for any such S.

Step 3: The Sensitivity Argument

The sensitivity of a Boolean function f at an input x is the number of bits in the input such that flipping the bit changes the value of $f(x)$. We will use the following result from the theory of Boolean functions:

For a random Boolean function, the average sensitivity over all inputs is proportional to the number of input bits, i.e., it is of order n. Furthermore, for any partition of the input bits, if the size of the subset S is between $2n/3$ and $3n/4$, the communication complexity $C(f, S)$ is at least a constant fraction of n with high probability.

Step 4: Lower Bound on Communication Complexity

For a random Boolean function f, the communication complexity $C(f, S)$ is lower-bounded by a function of the sensitivity of f with respect to the partition defined by S. Specifically, the communication complexity is proportional to the number of bits in S that affect the output of the function. Since for most random functions, the sensitivity is proportional to n, we conclude that

$$C(f, S) \geq \epsilon n$$

for some universal constant $\epsilon > 0$, where the constant ϵ depends on the function f but is independent of the specific choice of S within the specified range.

Step 5: Conclusion

Thus, for most functions f, the communication complexity $C(f, S)$ satisfies the inequality

$$C(f, S) \geq \epsilon n$$

for every set S with $2n/3 \leq |S| \leq 3n/4$. This completes the proof.

□

B.9.3 Claude

Proof. We prove this theorem by a counting argument combined with a probabilistic method approach.

First, we establish the number of functions with low communication complexity. For a fixed partition $(S, \bar{S})$ where $S \subseteq [2n]$ with $2n/3 \leq |S| \leq 3n/4$, if $C(f, S) < \epsilon n$ for some $\epsilon > 0$ to be determined, then there exists a communication protocol using fewer than ϵn bits.

Any deterministic communication protocol with communication cost c partitions the input space $\{0,1\}^{2n}$ into at most 2^{c+1} combinatorial rectangles (since each bit communicated at most doubles the number of rectangles). Each rectangle $R = A \times B$ where $A \subseteq \{0,1\}^{|S|}$ and $B \subseteq \{0,1\}^{2n-|S|}$ must be monochromatic under f.

Therefore, if $C(f, S) < \epsilon n$, the function f is constant on each of at most $2^{\epsilon n+1}$ rectangles. The number of such functions is at most $2^{2^{\epsilon n+1}}$ since we can assign each rectangle a value in $\{0,1\}$ independently.

Now we count the number of problematic partitions. The number of subsets $S \subseteq [2n]$ with $2n/3 \leq |S| \leq 3n/4$ is at most $\sum_{k=\lceil 2n/3 \rceil}^{\lfloor 3n/4 \rfloor} \binom{2n}{k} \leq 2^{2n}$ by the standard bound on binomial coefficients.

The number of functions $f : \{0,1\}^{2n} \to \{0,1\}$ that have $C(f, S) < \epsilon n$ for at least one S in the specified range is at most

$$2^{2n} \cdot 2^{2^{\epsilon n+1}}$$

by the union bound.

The total number of functions $f : \{0,1\}^{2n} \to \{0,1\}$ is $2^{2^{2n}}$.

For most functions to satisfy $C(f, S) \geq \epsilon n$ for all S with $2n/3 \leq |S| \leq 3n/4$, we need

$$2^{2n} \cdot 2^{2^{\epsilon n+1}} < 2^{2^{2n}}$$

Taking logarithms:

$$2n + 2^{\epsilon n+1} < 2^{2n}$$

For sufficiently large n, we have $2^{2n} \gg 2n$. We need $2^{\epsilon n+1} < 2^{2n}$, which gives us $\epsilon n + 1 < 2n$, or equivalently $\epsilon < 2 - \frac{1}{n}$.

However, we need a more careful analysis. Note that for $|S| \geq 2n/3$, we have $|\bar{S}| \leq n/3$. The key observation is that when the partition is unbalanced with $|\bar{S}| \leq n/3$, Bob has limited information.

By information-theoretic arguments, since Bob receives only $|\bar{S}| \leq n/3$ bits of input, and Alice must communicate enough information for Bob to determine f on all $2^{|S|} \geq 2^{2n/3}$ possible inputs Alice might have, the communication complexity must be at least $\Omega(n)$ for a random function.

More precisely, for a random function, the expected communication complexity is at least $|S| - |\bar{S}| - O(\log n) \geq 2n/3 - n/3 - O(\log n) = n/3 - O(\log n)$.

Therefore, we can take $\epsilon = 1/12$ (or any constant less than $1/3$), and by the probabilistic method, the fraction of functions violating the condition is exponentially small in n. Thus, for most functions, $C(f, S) \geq \epsilon n$ for all S with $2n/3 \leq |S| \leq 3n/4$. □

B.9.4 Gemini

Proof. Let $\mathcal{F}$ be the set of all boolean functions $f : \{0,1\}^{2n} \to \{0,1\}$. The size of this set is $|\mathcal{F}| = 2^{2^{2n}}$. We aim to show that for a sufficiently small constant $\epsilon > 0$, the fraction of functions f satisfying $C(f, S) < \epsilon n$ for any valid subset S tends to zero as $n \to \infty$.

Fix a subset $S \subseteq [2n]$ satisfying $2n/3 \leq |S| \leq 3n/4$. Let $\bar{S} = [2n] \setminus S$ be the set of indices assigned to Bob. Let $N = 2^{|S|}$ and $M = 2^{|\bar{S}|}$. The communication matrix M_f associated with f and the partition $(S, \bar{S})$ has dimensions $N \times M$.

A deterministic communication protocol of cost at most c can be modeled by a binary tree of depth at most c. 1. The number of binary trees of depth at most c is bounded by 4^{2^c}. 2. Each internal node v is owned by either Alice or Bob. If owned by Alice, the edge taken is determined by a function $a_v : \{0,1\}^{|S|} \to \{0,1\}$. If owned by Bob, it is determined by $b_v : \{0,1\}^{|\bar{S}|} \to \{0,1\}$. 3. Each leaf is labeled with an output value in $\{0,1\}$.

The number of distinct functions a_v is $2^{2^{|S|}} = 2^N$, and the number of distinct functions b_v is $2^{2^{|\bar{S}|}} = 2^M$. Since $|S| \leq 3n/4$ and $|\bar{S}| = 2n - |S| \geq 5n/4$, we have $N < M$. Thus, the number of choices for a function at any internal node is bounded by $2 \cdot 2^M$. The number of nodes in the tree is at most 2^c. Therefore, the number of distinct protocols of depth c, denoted by $\mathcal{P}_c$, is bounded by:

$$|\mathcal{P}_c| \leq 4^{2^c} \cdot \left(2 \cdot 2^M\right)^{2^c} \cdot 2^{2^c} = 2^{2^c(M+O(1))}.$$

Since each protocol computes exactly one function, the number of functions f with $C(f, S) \leq c$ is at most $|\mathcal{P}_c|$.

We require $C(f, S) \geq \epsilon n$. Let $c = \epsilon n$. We compare $|\mathcal{P}_c|$ with the total number of functions $|\mathcal{F}| = 2^{NM}$.

$$\frac{|\{f : C(f, S) \leq \epsilon n\}|}{|\mathcal{F}|} \leq \frac{2^{2^{\epsilon n}(2^{|\bar{S}|}+O(1))}}{2^{2^{|S|} \cdot 2^{|\bar{S}|}}} = 2^{2^{|\bar{S}|}\left(2^{\epsilon n} - 2^{|S|}\right)+O(2^{\epsilon n})}.$$

To show this fraction is small, we analyze the exponent. We need $2^{\epsilon n} \ll 2^{|S|}$. Taking logarithms, we require $\epsilon n < |S|$. Given the constraint $|S| \geq 2n/3$, we can choose any $\epsilon < 2/3$. Let $\epsilon = 1/2$. Then,

$$2^{\epsilon n} - 2^{|S|} = 2^{n/2} - 2^{|S|} \leq 2^{n/2} - 2^{2n/3}.$$

For sufficiently large n, this term is dominated by $-2^{2n/3}$. Thus, the probability that a random function has low complexity for a *fixed* S is bounded by:

$$\mathbb{P}_f[C(f, S) \leq n/2] \leq 2^{-2^{|\bar{S}|} \cdot 2^{2n/3} \cdot (1-o(1))} \leq 2^{-2^{2n-o(n)}}.$$

(Note that $|\bar{S}| + 2n/3 \geq 5n/4 + 2n/3 > 2n$, so the exponent is extremely large).

Finally, we apply the union bound over all valid subsets S. The number of subsets S is bounded by 2^{2n}.

$$\mathbb{P}_f\left[\exists S : \frac{2n}{3} \le |S| \le \frac{3n}{4} \wedge C(f,S) < \frac{n}{2}\right] \le \sum_S \mathbb{P}_f[C(f,S) \le n/2] \le 2^{2n} \cdot 2^{-2^{2n-o(n)}}.$$

As $n \to \infty$, this probability tends to 0. Thus, for most functions f, $C(f,S) \ge \frac{1}{2}n$ for all such S. □

B.10 Problem 1.10

B.10.1 Problem Statement

Theorem B.10. *Give an explicit example of a function $f : \{0,1\}^{4n} \to \{0,1\}$ for which $C(f,S) \ge \Omega(n)$ for all sets S with $|S| \le 4n/3$. Hint: Consider functions of the type $f(A,B,x) = \mathsf{IP}(x_A, x_B)$, where here $A, B \subset [m]$ are disjoint subsets of size $m/10$, $x \in \{0,1\}^m$, and x_A, x_B are the projections of x onto the coordinates of A and B.*

B.10.2 GPT

Proof. Let $N = 4n$ and write an input $z \in \{0,1\}^{4n}$ as a pair (u,v) with $u, v \in \{0,1\}^{2n}$. Define

$$f(u,v) = \mathsf{IP}(u,v) = \left(\sum_{i=1}^{2n} u_i v_i\right) \bmod 2.$$

Fix an arbitrary set $S \subseteq [4n]$ with $|S| \le \frac{4n}{3}$. For each $i \in [2n]$, let p_i denote the pair of coordinates corresponding to (u_i, v_i), i.e. $p_i = \{i, 2n+i\} \subseteq [4n]$. Define

$$P = \{\, i \in [2n] : p_i \cap S \ne \emptyset \,\}.$$

Since each $i \in P$ satisfies $|p_i \cap S| \ge 1$ and the pairs p_i are disjoint, we have $|P| \le |S| \le \frac{4n}{3}$. Hence the set

$$Q = [2n] \setminus P = \{\, i \in [2n] : p_i \cap S = \emptyset \,\}$$

satisfies $|Q| = 2n - |P| \ge 2n - \frac{4n}{3} = \frac{2n}{3}$, and for every $i \in Q$ we have $p_i \subseteq [4n] \setminus S$.

Let $z^0 = (u^0, v^0)$ be the all-zero input, i.e. $u^0 = 0^{2n}$ and $v^0 = 0^{2n}$. Then $f(z^0) = 0$. Consider any partial assignment ρ that assigns values only to coordinates in $[4n] \setminus S$ and is consistent with z^0 (so every assigned coordinate is set to 0). Suppose that ρ is a 0-certificate for f at z^0, meaning that for every completion (u,v) consistent with ρ we have $f(u,v) = 0$. We claim that ρ must assign at least one coordinate from each pair p_i with $i \in Q$.

Assume for contradiction that there exists some $i \in Q$ such that ρ assigns neither coordinate in p_i, i.e. both u_i and v_i are left free by ρ. Define two completions $(u^{(0)}, v^{(0)})$ and $(u^{(1)}, v^{(1)})$ consistent with ρ as follows. For all coordinates assigned by ρ, set them according to ρ (hence to 0). For all other coordinates, set them to 0 except possibly at the pair (u_i, v_i): set $u_i^{(0)} = v_i^{(0)} = 0$ and set $u_i^{(1)} = v_i^{(1)} = 1$. Since $i \in Q$, both coordinates in p_i lie in $[4n] \setminus S$, and since ρ assigns neither, both completions are consistent with ρ.

Compute f on these completions. In $(u^{(0)}, v^{(0)})$, every product $u_j v_j$ equals 0, so $f(u^{(0)}, v^{(0)}) = 0$. In $(u^{(1)}, v^{(1)})$, we have $u_i^{(1)} v_i^{(1)} = 1$ and for all $j \ne i$ we still have $u_j^{(1)} v_j^{(1)} = 0$, hence

$$f(u^{(1)}, v^{(1)}) = \left(\sum_{j=1}^{2n} u_j^{(1)} v_j^{(1)}\right) \bmod 2 = 1.$$

Thus ρ cannot force f to be constantly 0 on its consistent completions, contradicting that ρ is a 0-certificate at z^0. Therefore, for every $i \in Q$, the partial assignment ρ must assign at least one coordinate of p_i.

Since the pairs p_i are disjoint, assigning at least one coordinate from each p_i with $i \in Q$ requires that ρ assign at least $|Q|$ coordinates. Consequently, every 0-certificate at z^0 that assigns only coordinates outside S has size at least $|Q| \ge \frac{2n}{3}$. Hence $C(f,S) \ge \frac{2n}{3}$ for every S with $|S| \le \frac{4n}{3}$, and in particular $C(f,S) \ge \Omega(n)$ for all such S. □

B.10.3 Claude

Proof. We construct an explicit function $f : \{0,1\}^{4n} \to \{0,1\}$ satisfying the required property.

Let $m = 4n$ and define the input space as $\{0,1\}^m$. We partition the m coordinates into three parts: coordinates corresponding to set A, coordinates corresponding to set B, and the remaining coordinates for x. Specifically, let $A, B \subset [m]$ be disjoint subsets with $|A| = |B| = m/10 = 2n/5$. Let the remaining coordinates form the set X with $|X| = m - 2|A| = m - m/5 = 4m/5 = 16n/5$.

For an input $w \in \{0,1\}^m$, we interpret w as specifying two subsets $A_w, B_w \subset X$ (encoded in the coordinates of A and B respectively) and a string $x \in \{0,1\}^{|X|}$ (in the remaining coordinates). However, to match the hint more directly, we use a simpler encoding.

Let us redefine more carefully. We set $m = 10n$ so that $m/10 = n$. We view the input as (a, b, x) where $a, b \in \{0,1\}^n$ and $x \in \{0,1\}^{8n}$. This gives total input length $n + n + 8n = 10n$. To match the requirement of $4n$ input bits, we instead set $m = 4n$ and $|A| = |B| = n$.

We define $f : \{0,1\}^{4n} \to \{0,1\}$ as follows. Partition the $4n$ input bits into two parts: the first $2n$ bits encode (a, b) where $a, b \in \{0,1\}^n$, and we define

$$f(a, b) = \mathsf{IP}(a, b) = \sum_{i=1}^{n} a_i b_i \pmod 2.$$

Now we prove that $C(f, S) \geq \Omega(n)$ for all sets S with $|S| \leq 4n/3$.

Consider any set $S \subseteq [4n]$ with $|S| \leq 4n/3$. Let $S_a = S \cap [n]$ be the coordinates in a, and $S_b = S \cap [n+1, 2n]$ be the coordinates in b. Note that $|S_a| + |S_b| \leq |S| \leq 4n/3$.

By the pigeonhole principle, either $|S_a| \leq 2n/3$ or $|S_b| \leq 2n/3$ (or both). Without loss of generality, assume $|S_a| \leq 2n/3$. Then there exist at least $n - 2n/3 = n/3$ coordinates in a that are not in S.

For the inner product function $\mathsf{IP}(a, b)$, it is well-known that any decision tree computing IP on n-bit inputs has depth at least n. More precisely, if we restrict to querying only coordinates in $S_a \cup S_b$, and $\min(|S_a|, |S_b|) < n$, then we cannot determine $\mathsf{IP}(a, b)$ with certainty.

Specifically, if $|S_a| < n$, there exists an index $i \in [n]$ with $i \notin S_a$. For any partial assignment to coordinates in S, we can find two complete inputs (a, b) and (a', b') that agree on S but differ on coordinate i of a and some coordinate j of b (with $j \notin S_b$ if $|S_b| < n$), such that $\mathsf{IP}(a, b) \neq \mathsf{IP}(a', b')$.

By standard arguments for communication complexity lower bounds for inner product, we have that $C(f, S) \geq \min(n - |S_a|, n - |S_b|) \geq n - 2n/3 = n/3 = \Omega(n)$.

Therefore, $f(a, b) = \mathsf{IP}(a, b)$ on inputs of length $4n$ (with the last $2n$ bits unused or set to zero) satisfies $C(f, S) \geq \Omega(n)$ for all $|S| \leq 4n/3$. □

B.10.4 Gemini

Proof. Let $N = 4n$. We construct a function $f : \{0,1\}^{4n} \to \{0,1\}$ as follows. Partition the input variables $z \in \{0,1\}^{4n}$ into two parts: $x \in \{0,1\}^{2n}$ and $y \in \{0,1\}^{2n}$. Let $m = 2n$ and let $k = \lfloor n/10 \rfloor$. We interpret the string y as an encoding of two disjoint subsets $A, B \subset \{1, \ldots, m\}$ with $|A| = |B| = k$. The number of such pairs of disjoint subsets is given by the multinomial coefficient $\binom{m}{k,k,m-2k}$. We have:

$$\log_2 \binom{2n}{k, k, 2n-2k} \approx 2n \left(H\left(\frac{1}{20}\right) + \frac{19}{20} H\left(\frac{1}{19}\right) \right) < 2n,$$

where H is the binary entropy function. Thus, there exists a surjective mapping from $\{0,1\}^{2n}$ to the set of pairs of disjoint subsets (A, B) of size k. For a given y, let (A_y, B_y) denote the corresponding sets.

Define the function $f(x, y)$ as the inner product of the projections of x onto A_y and B_y:

$$f(x, y) = \mathsf{IP}(x_{A_y}, x_{B_y}) = \sum_{j=1}^{k} x_{(A_y)_j} \cdot x_{(B_y)_j} \pmod 2,$$

where $(A_y)_j$ and $(B_y)_j$ denote the j-th elements of the sets A_y and B_y under the natural ordering of indices.

Consider any partition of the input indices into two sets S and $\bar{S}$ such that $|S| \leq 4n/3$. We analyze the communication complexity $C(f, S)$ where Alice holds the variables indexed by S and Bob holds the variables indexed by $\bar{S}$. Note that since $|S| \leq 4n/3$, we have $|\bar{S}| \geq 8n/3$.

Let $I_x = \{1, \ldots, 2n\}$ be the indices corresponding to x, and $I_y = \{2n+1, \ldots, 4n\}$ be the indices corresponding to y. Define $S_x = S \cap I_x$ and $\bar{S}_x = \bar{S} \cap I_x$. We distinguish two cases based on the distribution of x-variables.

Case 1: $|S_x| \geq k$ and $|\bar{S}_x| \geq k$. In this case, both Alice and Bob hold at least k variables of x. We can fix the variables in y to a constant value y^* such that the corresponding sets A_{y^*} and B_{y^*} satisfy $A_{y^*} \subseteq S_x$ and $B_{y^*} \subseteq \bar{S}_x$. Such a choice is possible because A and B are disjoint and we can choose any k indices from S_x to form A and any k indices from $\bar{S}_x$ to form B. With y fixed to y^*, the function f reduces to:

$$f(x, y^*) = \sum_{j=1}^{k} x_{(A_{y^*})_j} \cdot x_{(B_{y^*})_j} \pmod 2.$$

Since $A_{y^*} \subseteq S_x$, Alice holds all variables x_i for $i \in A_{y^*}$. Since $B_{y^*} \subseteq \bar{S}_x$, Bob holds all variables x_i for $i \in B_{y^*}$. This is exactly the Inner Product function on k bits, denoted IP_k. It is a standard result that $C(\mathsf{IP}_k) \geq k$. Since $k = \lfloor n/10 \rfloor = \Omega(n)$, we have $C(f, S) \geq \Omega(n)$.

Case 2: $|S_x| < k$ or $|\bar{S}_x| < k$. First, consider $|\bar{S}_x| < k$. Since the total number of x variables is $2n$ and $k \approx 0.1n$, this would imply $|S_x| > 1.9n$. However, we are given $|S| \leq 4n/3 \approx 1.33n$. Thus, it is impossible for $|S_x| > 1.9n$. Therefore, we must have $|S_x| < k$. This implies that Bob holds the vast majority of x variables: $|\bar{S}_x| = 2n - |S_x| > 2n - k$.

Now we consider the dependency on y. The function f essentially requires selecting specific bits of x based on y. If $|S|$ is small (e.g., $|S| < \epsilon n$), Alice can simply send all her bits to Bob, resulting in complexity $|S|$. The theorem statement implies we are looking for a lower bound of $\Omega(n)$, which implicitly assumes $|S| = \Omega(n)$. Assuming $|S| \geq \delta n$ for some constant δ, we proceed.

Since $|S_x|$ is small, S must contain a significant number of y variables. Specifically, if $|S| \approx n$, then $|S_y|$ is large. We reduce to the *Indexing* problem. Bob holds almost all of x (the "database"). Alice holds a part of y (part of the "index"). Let us fix the variables in $\bar{S}_y$ (Bob's part of y) and the variables in S_x (Alice's part of x) to constants. Alice still holds variables in S_y which determine the sets A, B partially. Bob holds variables in $\bar{S}_x$ which include the target bits for the inner product. Because the mapping $y \to (A, B)$ is surjective and "rich", the entropy of the pair (A, B) given $\bar{S}_y$ is high. Specifically, for a random x, computing $f(x, y)$ requires information about (A, B) roughly proportional to the description length of the subsets. Given that Bob holds x, he needs to know A and B to compute the sum. Alice holds the information defining A and B. The communication complexity is lower bounded by the necessary information transfer regarding y, which is $\Omega(|S_y|)$. If $|S| = \Omega(n)$ and $|S_x|$ is small, then $|S_y| = \Omega(n)$. Thus, $C(f, S) \geq \Omega(n)$.

Combining both cases, for any partition S with $|S| \leq 4n/3$ (and implicitly $|S| = \Omega(n)$), the communication complexity is $\Omega(n)$. □

B.11 Problem 1.11

B.11.1 Problem Statement

Theorem B.11. *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function with the property that for every $x \in \{0,1\}^n$, there are exactly t choices for y with $f(x, y) = 1$, and for every $y \in \{0,1\}^n$, there are exactly t choices for x with $f(x, y) = 1$. Then there exist permutations $\pi_1, \ldots, \pi_t$ of $\{0,1\}^n$ such that $f(x, y) = 1$ if and only if $y = \pi_r(x)$ for some r, and f must admit a 0-cover of size at most $(2n)^t$.*

B.11.2 GPT

Proof. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function satisfying the given conditions: for every $x \in \{0,1\}^n$, there are exactly t choices for y such that $f(x, y) = 1$, and for every $y \in \{0,1\}^n$, there are exactly t choices for x such that $f(x, y) = 1$. We aim to prove the existence of permutations $\pi_1, \ldots, \pi_t$ of $\{0,1\}^n$ such that $f(x, y) = 1$ if and only if $y = \pi_r(x)$ for some r, and that f admits a 0-cover of size at most $(2n)^t$.

Step 1: Defining the permutations First, fix an arbitrary $x_0 \in \{0,1\}^n$. The condition that for each $x \in \{0,1\}^n$, there are exactly t choices for y such that $f(x,y) = 1$ implies that for x_0, there exist exactly t values $y_1, y_2, \ldots, y_t \in \{0,1\}^n$ such that $f(x_0, y_i) = 1$ for each $i \in \{1, 2, \ldots, t\}$.

Next, define for each $x \in \{0,1\}^n$ the set of corresponding y values where $f(x,y) = 1$, i.e., the set $S_x = \{y \in \{0,1\}^n : f(x,y) = 1\}$. By the given condition, $|S_x| = t$ for all $x \in \{0,1\}^n$. For each x, label the elements of S_x as $y_1(x), y_2(x), \ldots, y_t(x)$.

Now, we will construct the permutations $\pi_1, \pi_2, \ldots, \pi_t$. For each $r \in \{1, 2, \ldots, t\}$, define π_r as the permutation of $\{0,1\}^n$ that maps $x \in \{0,1\}^n$ to the r-th element of the set S_x. Specifically, let $\pi_r(x) = y_r(x)$. Thus, for each $x \in \{0,1\}^n$, we have a collection of t distinct permutations $\pi_1, \pi_2, \ldots, \pi_t$ such that $f(x,y) = 1$ if and only if $y = \pi_r(x)$ for some $r \in \{1, 2, \ldots, t\}$.

Step 2: Verifying the permutations We now verify that these permutations $\pi_1, \pi_2, \ldots, \pi_t$ satisfy the condition that $f(x,y) = 1$ if and only if $y = \pi_r(x)$ for some $r \in \{1, 2, \ldots, t\}$.

By the construction of the permutations, for each $x \in \{0,1\}^n$ and each $r \in \{1, 2, \ldots, t\}$, we have $y_r(x) \in S_x$. Therefore, for each x, there exist exactly t values of y such that $f(x,y) = 1$, namely $y_1(x), y_2(x), \ldots, y_t(x)$. These are precisely the images of x under the permutations $\pi_1, \pi_2, \ldots, \pi_t$. Hence, we have $f(x,y) = 1$ if and only if $y = \pi_r(x)$ for some $r \in \{1, 2, \ldots, t\}$.

Step 3: Constructing a 0-cover Finally, we need to prove that f admits a 0-cover of size at most $(2n)^t$. A 0-cover for a function f is a set of pairs (x,y) such that $f(x,y) = 0$ for all other pairs (x', y') not in the cover.

Consider the set of pairs $(x, \pi_r(x))$ for each $x \in \{0,1\}^n$ and each $r \in \{1, 2, \ldots, t\}$. Since there are t such pairs for each x, the total number of pairs is at most $t \times 2^n$. Since 2^n is the size of $\{0,1\}^n$, the total number of pairs is at most $(2n)^t$. Hence, we conclude that the size of the 0-cover is at most $(2n)^t$.

Thus, we have shown that there exist permutations $\pi_1, \ldots, \pi_t$ of $\{0,1\}^n$ such that $f(x,y) = 1$ if and only if $y = \pi_r(x)$ for some r, and that f admits a 0-cover of size at most $(2n)^t$.

□

B.11.3 Claude

Proof. We prove both parts of the theorem separately.

Part 1: Existence of permutations.

Define the relation $R \subseteq \{0,1\}^n \times \{0,1\}^n$ by $(x,y) \in R$ if and only if $f(x,y) = 1$. By hypothesis, for every $x \in \{0,1\}^n$, there are exactly t elements y with $(x,y) \in R$, and for every $y \in \{0,1\}^n$, there are exactly t elements x with $(x,y) \in R$.

Consider the bipartite graph $G = (X \cup Y, E)$ where $X = Y = \{0,1\}^n$ and $(x,y) \in E$ if and only if $f(x,y) = 1$. This is a t-regular bipartite graph on 2^n vertices per side.

By König's theorem and Hall's marriage theorem, a t-regular bipartite graph can be decomposed into t perfect matchings. Specifically, since the graph is t-regular, we can apply edge-coloring results for regular bipartite graphs to obtain a decomposition $E = M_1 \cup M_2 \cup \cdots \cup M_t$ where each M_r is a perfect matching.

For each $r \in \{1, \ldots, t\}$, the perfect matching M_r defines a bijection $\pi_r : \{0,1\}^n \to \{0,1\}^n$ where $\pi_r(x) = y$ if and only if $(x,y) \in M_r$. Since M_r is a perfect matching, π_r is indeed a permutation of $\{0,1\}^n$.

By construction, $f(x,y) = 1$ if and only if $(x,y) \in E = \bigcup_{r=1}^{t} M_r$, which occurs if and only if $(x,y) \in M_r$ for some $r \in \{1, \ldots, t\}$, which is equivalent to $y = \pi_r(x)$ for some r.

Part 2: Existence of 0-cover.

A 0-cover is a set of rectangles (products of subsets) that covers all pairs (x,y) with $f(x,y) = 0$.

For each $r \in \{1, \ldots, t\}$ and each coordinate $i \in \{1, \ldots, n\}$, define the rectangle $R_{r,i,b}$ for $b \in \{0,1\}$ by

$$R_{r,i,b} = \{(x,y) : x_i = b \text{ and } \pi_r(x)_i \neq y_i\}.$$

We claim that the collection $\mathcal{C} = \{R_{r,i,b} : r \in \{1, \ldots, t\}, i \in \{1, \ldots, n\}, b \in \{0,1\}\}$ forms a 0-cover of size $2nt$.

First, observe that $|\mathcal{C}| = t \cdot n \cdot 2 = 2nt \leq (2n)^t$ for $t \geq 1$.

Now we verify that $\mathcal{C}$ covers all zeros of f. Let (x, y) be such that $f(x, y) = 0$. Then $y \neq \pi_r(x)$ for all $r \in \{1, \ldots, t\}$.

Since $y \neq \pi_1(x)$, there exists some coordinate $i_1 \in \{1, \ldots, n\}$ such that $y_{i_1} \neq \pi_1(x)_{i_1}$. Let $b_1 = x_{i_1}$. Then $(x, y) \in R_{1,i_1,b_1}$ since $x_{i_1} = b_1$ and $\pi_1(x)_{i_1} \neq y_{i_1}$.

Therefore, every pair (x, y) with $f(x, y) = 0$ is covered by at least one rectangle in $\mathcal{C}$.

Conversely, if $(x, y) \in R_{r,i,b}$ for some r, i, b, then $x_i = b$ and $\pi_r(x)_i \neq y_i$, which implies $\pi_r(x) \neq y$, hence $f(x, y) = 0$ (since $f(x, y) = 1$ would require $y = \pi_s(x)$ for some s, but if $s = r$ we have a contradiction).

Thus $\mathcal{C}$ is a valid 0-cover of size $2nt \leq (2n)^t$. □

B.11.4 Gemini

Proof. Let $V = \{0, 1\}^n$. We define a bipartite graph $G = (L, R, E)$ where $L = R = V$. An edge $(x, y) \in L \times R$ exists in E if and only if $f(x, y) = 1$.

The problem statement guarantees that for every $x \in L$, the degree $\deg(x) = |\{y : f(x, y) = 1\}| = t$. Similarly, for every $y \in R$, the degree $\deg(y) = |\{x : f(x, y) = 1\}| = t$. Thus, G is a t-regular bipartite graph.

A well-known corollary of Hall's Marriage Theorem states that every k-regular bipartite graph with $k \geq 1$ admits a perfect matching. Since G is t-regular, if $t > 0$, there exists a perfect matching $M_1 \subseteq E$. We may iteratively decompose the edge set. Let $E_1 = E$. Since (L, R, E_1) is t-regular, it contains a perfect matching M_1. We define $E_2 = E_1 \setminus M_1$. The graph (L, R, E_2) is $(t - 1)$-regular. Proceeding inductively, we obtain t disjoint perfect matchings $M_1, \ldots, M_t$ such that $E = \bigcup_{r=1}^{t} M_r$. If $t = 0$, the edge set is empty and the statement holds vacuously.

For each $r \in \{1, \ldots, t\}$, the perfect matching M_r defines a bijection from L to R. Let $\pi_r : \{0, 1\}^n \to \{0, 1\}^n$ be the permutation defined by $y = \pi_r(x)$ if and only if $(x, y) \in M_r$. Since the matchings partition the edges where $f(x, y) = 1$, we have:

$$f(x, y) = 1 \iff \exists r \in \{1, \ldots, t\} \text{ such that } y = \pi_r(x).$$

This establishes the first part of the theorem.

Next, we construct a 0-cover for f. A 0-cover is a collection of combinatorial rectangles $A \times B \subseteq \{0, 1\}^n \times \{0, 1\}^n$ such that every pair (x, y) with $f(x, y) = 0$ is contained in at least one rectangle in the collection, and no pair with $f(x, y) = 1$ is contained in any rectangle in the collection.

The condition $f(x, y) = 0$ is equivalent to the negation of the condition for $f(x, y) = 1$. Thus:

$$f(x, y) = 0 \iff \forall r \in \{1, \ldots, t\}, y \neq \pi_r(x).$$

Let $S_r = \{(x, y) \in \{0, 1\}^n \times \{0, 1\}^n : y \neq \pi_r(x)\}$. The set of zero-inputs of f is $f^{-1}(0) = \bigcap_{r=1}^{t} S_r$.

We first analyze the size of a cover for a single set S_r. The condition $y \neq \pi_r(x)$ holds if and only if there exists an index $i \in \{1, \ldots, n\}$ such that the i-th bit of y differs from the i-th bit of $\pi_r(x)$. Let $(\pi_r(x))_i$ denote the i-th bit of the vector $\pi_r(x)$. We have:

$$y \neq \pi_r(x) \iff \bigvee_{i=1}^{n} \left((y_i = 0 \wedge (\pi_r(x))_i = 1) \vee (y_i = 1 \wedge (\pi_r(x))_i = 0) \right).$$

For a fixed r and i, define the sets:

$$U_{r,i,0} = \{x \in \{0, 1\}^n : (\pi_r(x))_i = 0\}, \quad U_{r,i,1} = \{x \in \{0, 1\}^n : (\pi_r(x))_i = 1\}.$$

Define also the sets based on y:

$$V_{i,0} = \{y \in \{0, 1\}^n : y_i = 0\}, \quad V_{i,1} = \{y \in \{0, 1\}^n : y_i = 1\}.$$

The condition $y_i \neq (\pi_r(x))_i$ corresponds to the union of two rectangles:

$$R_{r,i}^{(0)} = U_{r,i,1} \times V_{i,0} \quad \text{and} \quad R_{r,i}^{(1)} = U_{r,i,0} \times V_{i,1}.$$

Note that for any $(x, y) \in R_{r,i}^{(b)}$, we have $y \neq \pi_r(x)$, so $f(x, y)$ is not necessarily 0, but $(x, y) \in S_r$. The set S_r is exactly the union of these rectangles over all i:

$$S_r = \bigcup_{i=1}^{n} \left(R_{r,i}^{(0)} \cup R_{r,i}^{(1)} \right).$$

Thus, S_r admits a cover $\mathcal{C}_r$ consisting of exactly $2n$ rectangles.

The set of inputs where $f(x, y) = 0$ is the intersection of these sets:

$$f^{-1}(0) = \bigcap_{r=1}^{t} S_r = \bigcap_{r=1}^{t} \left(\bigcup_{R \in \mathcal{C}_r} R \right).$$

By the distributive law of sets, this intersection can be rewritten as a union of intersections:

$$f^{-1}(0) = \bigcup_{(R_1, \ldots, R_t) \in \mathcal{C}_1 \times \cdots \times \mathcal{C}_t} \left(\bigcap_{r=1}^{t} R_r \right).$$

The intersection of rectangles is itself a rectangle (or empty). Specifically, if $R_r = A_r \times B_r$, then $\bigcap_{r=1}^{t} R_r = (\bigcap_{r=1}^{t} A_r) \times (\bigcap_{r=1}^{t} B_r)$. Furthermore, if $(x, y) \in \bigcap_{r=1}^{t} S_r$, then $y \neq \pi_r(x)$ for all r, which implies $f(x, y) = 0$. Therefore, each intersection rectangle consists entirely of 0-inputs.

The number of such intersection rectangles is bounded by the product of the sizes of the covers $\mathcal{C}_r$. Since $|\mathcal{C}_r| = 2n$ for each r, the total number of rectangles in the 0-cover is at most:

$$\prod_{r=1}^{t} |\mathcal{C}_r| = (2n)^t.$$

Thus, f admits a 0-cover of size at most $(2n)^t$. □

B.12 Problem 1.12

B.12.1 Problem Statement

Theorem B.12. *This exercise shows that an analogue of Theorem 1.36 does not hold for* partial functions. *These are functions that are defined on a subset of the domain. Consider the partial function* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$*, where the input to each party is interpreted as two* $n/2$ *bit strings, defined by*

$$f(x, x', y, y') = \begin{cases} 1 & \text{if } x = y \text{ and } x' \neq y', \\ 0 & \text{if } x \neq y \text{ and } x' = y'. \end{cases}$$

Show that there are $2n$ *monochromatic rectangles that cover the domain of* f*. A rectangle is considered monochromatic if* f *takes on the same value on any two inputs of the rectangle where* f *has been defined. Show that the communication complexity of* f *is at least* $\Omega(n)$*.*

B.12.2 GPT

Proof. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a partial function defined by

$$f(x, x', y, y') = \begin{cases} 1 & \text{if } x = y \text{ and } x' \neq y', \\ 0 & \text{if } x \neq y \text{ and } x' = y'. \end{cases}$$

We aim to show that there are $2n$ monochromatic rectangles that cover the domain of f, and that the communication complexity of f is at least $\Omega(n)$.

Step 1: Understanding the domain and the function The domain of the function f is $\{0,1\}^n \times \{0,1\}^n$, which consists of all pairs of n-bit strings. Thus, the total number of inputs to f is $2^n \times 2^n = 2^{2n}$.

The function f depends on four variables: (x, x', y, y'), where $x, y \in \{0,1\}^n$ and $x', y' \in \{0,1\}^n$. We can interpret x, y as two n-bit strings, and x', y' as two additional n-bit strings.

Step 2: Characterizing the behavior of f The function f has the following two cases:

1. $f(x, x', y, y') = 1$ if $x = y$ and $x' \neq y'$. 2. $f(x, x', y, y') = 0$ if $x \neq y$ and $x' = y'$.

The key observation here is that f is defined piecewise based on the equality and inequality conditions between the components of the inputs (x, x') and (y, y'). Specifically:

- $f(x, x', y, y') = 1$ if $x = y$ and $x' \neq y'$. - $f(x, x', y, y') = 0$ if $x \neq y$ and $x' = y'$.

Thus, the function assigns different values based on whether the pair of inputs (x, y) are equal or not, and similarly, based on the comparison between (x', y').

Step 3: Identifying the monochromatic rectangles A rectangle in the domain of f is a subset of inputs where the function f takes the same value for all points in the subset. We now show that there are $2n$ monochromatic rectangles that cover the entire domain.

To construct a monochromatic rectangle, consider the following:

- For each $x \in \{0, 1\}^n$, there is a corresponding set of inputs where $x = y$. For each such x, there are 2^n possible values for x', and 2^n possible values for y'. - We partition the domain into subsets based on the values of x and x', and similarly for y and y'. For each fixed x and x', we have 2^n possible values for y and y', forming rectangles.

In total, there are $2n$ such rectangles, corresponding to the combinations of $x = y$ and $x' \neq y'$, and $x \neq y$ and $x' = y'$. These rectangles are monochromatic because f takes the same value on all inputs within each rectangle.

Step 4: Communication complexity lower bound To analyze the communication complexity of f, we consider the communication complexity of the following tasks:

- Alice knows x, x' and must send information to Bob, who knows y, y', so that Bob can compute $f(x, x', y, y')$. - We observe that to determine f, Alice and Bob must communicate enough information to resolve the conditions $x = y$, $x' \neq y'$, and $x \neq y$, $x' = y'$. Since these conditions depend on n-bit strings, at least n bits of communication are required.

Hence, the communication complexity of f is at least $\Omega(n)$, as it takes at least n bits of information to resolve whether $x = y$ or $x \neq y$ and whether $x' = y'$ or $x' \neq y'$.

□

B.12.3 Claude

Proof. We prove both parts of the theorem.

Part 1: Existence of $2n$ monochromatic rectangles covering the domain of f.

First, observe that the domain of f consists of all tuples (x, x', y, y') where either $x = y$ and $x' \neq y'$, or $x \neq y$ and $x' = y'$. Here $x, y \in \{0, 1\}^{n/2}$ and $x', y' \in \{0, 1\}^{n/2}$.

For each $a \in \{0, 1\}^{n/2}$, define the rectangle $R_a^{(1)} = \{a\} \times \{0, 1\}^{n/2} \times \{a\} \times \{0, 1\}^{n/2}$. On this rectangle, whenever f is defined, we have $x = y = a$, which means f is defined precisely when $x' \neq y'$, and in all such cases $f(a, x', a, y') = 1$. Thus $R_a^{(1)}$ is monochromatic with value 1.

Similarly, for each $b \in \{0, 1\}^{n/2}$, define the rectangle $R_b^{(0)} = \{0, 1\}^{n/2} \times \{b\} \times \{0, 1\}^{n/2} \times \{b\}$. On this rectangle, whenever f is defined, we have $x' = y' = b$, which means f is defined precisely when $x \neq y$, and in all such cases $f(x, b, y, b) = 0$. Thus $R_b^{(0)}$ is monochromatic with value 0.

We have $2^{n/2}$ rectangles of type $R_a^{(1)}$ and $2^{n/2}$ rectangles of type $R_b^{(0)}$, giving a total of $2 \cdot 2^{n/2} = 2^{n/2+1}$ rectangles. However, the problem states there are $2n$ such rectangles, which suggests we should interpret the input differently or that n refers to the total number of bits per party.

Let us reinterpret: if each party receives n bits total (not $2n$ bits), then x, x' together form n bits where x is the first $n/2$ bits and x' is the last $n/2$ bits. Then we have $2^{n/2}$ choices for each of a and b, giving $2 \cdot 2^{n/2}$ rectangles. For this to equal $2n$, we need $2^{n/2} = n$, which holds when $n = 2$ or $n = 4$.

The construction shows that the domain can be covered by $2 \cdot 2^{n/2}$ monochromatic rectangles.

Part 2: Communication complexity is at least $\Omega(n)$.

To prove a lower bound on communication complexity, we use the fact that any deterministic protocol partitions the input space into monochromatic rectangles, one for each leaf of the protocol tree. The number of leaves is at least the minimum number of monochromatic rectangles needed to cover the domain where f outputs distinct values.

Consider the set $S_1 = \{(a, x', a, y') : a \in \{0,1\}^{n/2}, x', y' \in \{0,1\}^{n/2}, x' \neq y'\}$ where $f = 1$, and $S_0 = \{(x, b, y, b) : x, y \in \{0,1\}^{n/2}, b \in \{0,1\}^{n/2}, x \neq y\}$ where $f = 0$.

Any monochromatic rectangle $R = A \times B$ can contain points from only one of S_1 or S_0. For a rectangle to contain a point (a, x', a, y') from S_1, it must have both (a, x') and (a, y') in A. If A contains (a, x') and (a', x'') with $a \neq a'$, then R would contain $(a, x') \times (a', y'')$ which could be in S_0 if $x' = y''$, creating a conflict.

By a fooling set argument, consider pairs $\{(a, 0^{n/2}), (a, 1^{n/2})\}$ for each $a \in \{0,1\}^{n/2}$. These $2^{n/2+1}$ inputs require distinct rectangles in any partition, implying communication complexity at least $\log_2(2^{n/2+1}) = n/2 + 1 = \Omega(n)$.

Therefore, the communication complexity of f is at least $\Omega(n)$. □

B.12.4 Gemini

Proof. **Part 1: Monochromatic Cover**

We construct a set of monochromatic rectangles that cover the domain of f. The domain is $\text{Dom}(f) = f^{-1}(1) \cup f^{-1}(0)$, where:

$$f^{-1}(1) = \{(x, x', y, y') \mid x = y \text{ and } x' \neq y'\}$$

$$f^{-1}(0) = \{(x, x', y, y') \mid x \neq y \text{ and } x' = y'\}$$

Recall that $x, x', y, y' \in \{0,1\}^{n/2}$.

To cover $f^{-1}(1)$, consider the condition $x' \neq y'$. Since x' and y' differ, they must differ in at least one bit position $j \in \{1, \ldots, n/2\}$. For each j and each bit value $b \in \{0,1\}$, define the rectangle:

$$R^1_{j,b} = \{(x, x') \mid x'_j = b\} \times \{(y, y') \mid y'_j = 1 - b\}$$

If $((x, x'), (y, y')) \in R^1_{j,b} \cap \text{Dom}(f)$, then $x'_j = b$ and $y'_j = 1 - b$, so $x'_j \neq y'_j$, which implies $x' \neq y'$. If f is defined, we cannot have $f = 0$ (since $f = 0 \implies x' = y'$). Thus, if defined, f must be 1. Hence, $R^1_{j,b}$ is a 1-monochromatic rectangle. The union $\bigcup_{j,b} R^1_{j,b}$ covers all inputs where $x' \neq y'$, and thus covers $f^{-1}(1)$. There are $2 \cdot (n/2) = n$ such rectangles.

To cover $f^{-1}(0)$, consider the condition $x \neq y$. Since x and y differ, they must differ in at least one bit position $i \in \{1, \ldots, n/2\}$. For each i and $b \in \{0,1\}$, define the rectangle:

$$R^0_{i,b} = \{(x, x') \mid x_i = b\} \times \{(y, y') \mid y_i = 1 - b\}$$

If $((x, x'), (y, y')) \in R^0_{i,b} \cap \text{Dom}(f)$, then $x_i \neq y_i$, so $x \neq y$. If f is defined, we cannot have $f = 1$ (since $f = 1 \implies x = y$). Thus, if defined, f must be 0. Hence, $R^0_{i,b}$ is a 0-monochromatic rectangle. The union $\bigcup_{i,b} R^0_{i,b}$ covers all inputs where $x \neq y$, and thus covers $f^{-1}(0)$. There are $2 \cdot (n/2) = n$ such rectangles.

The total number of rectangles is $n + n = 2n$.

Part 2: Communication Complexity Lower Bound

We show that $D(f) = \Omega(n)$ by reducing the Equality problem to f. Let $EQ : \{0,1\}^{n/2} \times \{0,1\}^{n/2} \to \{0,1\}$ be the equality function, defined by $EQ(u, v) = 1$ if $u = v$ and 0 if $u \neq v$. It is well known that $D(EQ) \geq n/2$.

Let P be a deterministic protocol computing f. We construct a protocol for EQ. Alice and Bob receive inputs u and v respectively. Alice constructs the input $(u, 0^{n/2})$ for P. Bob constructs two inputs for P: $(v, 1^{n/2})$ and $(v, 0^{n/2})$.

Bob runs the protocol P with Alice twice: 1. Run 1: Alice uses $(u, 0^{n/2})$, Bob uses $(v, 1^{n/2})$. Let the output be o_1.

- If $u = v$, the input to f is $x = u, y = v \implies x = y$, and $x' = 0^{n/2}, y' = 1^{n/2} \implies x' \neq y'$. This is in $f^{-1}(1)$, so P must output $o_1 = 1$.
- If $u \neq v$, the input has $x \neq y$ and $x' \neq y'$. This is undefined; o_1 can be 0 or 1.

2. Run 2: Alice uses $(u, 0^{n/2})$, Bob uses $(v, 0^{n/2})$. Let the output be o_2.

- If $u \neq v$, the input to f is $x = u, y = v \implies x \neq y$, and $x' = 0^{n/2}, y' = 0^{n/2} \implies x' = y'$. This is in $f^{-1}(0)$, so P must output $o_2 = 0$.
- If $u = v$, the input has $x = y$ and $x' = y'$. This is undefined; o_2 can be 0 or 1.

Now, consider the combined output (o_1, o_2).

- If $u = v$, we must have $o_1 = 1$. The value of o_2 is unconstrained by the function definition, but determined by the protocol.
- If $u \neq v$, we must have $o_2 = 0$. The value of o_1 is unconstrained.

For the protocol P to be valid for f, it must be consistent. However, to solve EQ, we need to distinguish $u = v$ from $u \neq v$. Consider the function $g(u, v)$ computed by the protocol P on the fixed auxiliary inputs $x' = 0, y' = 1$. That is, $g(u, v) = P((u, 0), (v, 1))$. Consider the function $h(u, v)$ computed by P on $x' = 0, y' = 0$. That is, $h(u, v) = P((u, 0), (v, 0))$. From the conditions above:

- If $u = v$, $g(u, v) = 1$.
- If $u \neq v$, $h(u, v) = 0$.

Any deterministic protocol P defines specific functions g and h. The communication cost of P is at least the cost of computing g (or h). Suppose $D(f) < n/2 - 1$. Then there exists a protocol P with cost $< n/2 - 1$. This protocol defines a function $g(u, v)$ such that $g(u, u) = 1$ for all u. It also defines $h(u, v)$ such that $h(u, v) = 0$ for all $u \neq v$. However, g and h are derived from the *same* protocol P where only Bob's input changes. Specifically, Bob can determine both $g(u, v)$ and $h(u, v)$ by simulating P twice (Alice's behavior depends only on u). Let $A(u)$ be the message transcript sent by Alice. Since the cost is small, Alice partitions the set of inputs $\{0, 1\}^{n/2}$ into sets $U_1, \ldots, U_k$ with $k < 2^{n/2-1}$. By the pigeonhole principle, there exists a set U_i containing at least 3 distinct elements u_1, u_2, u_3. For these inputs, Alice sends the same messages. Bob's output depends only on his input v and the transcript. Thus, for a fixed v, $g(u, v)$ is constant for all $u \in U_i$. Similarly $h(u, v)$ is constant for all $u \in U_i$. Let $v = u_1$. Since $u_1 \in U_i$, $g(u_1, u_1) = 1$. Since $g(u, u_1)$ is constant for $u \in U_i$, $g(u_2, u_1) = 1$. Since $u_2 \neq u_1$, we must have $h(u_2, u_1) = 0$. Since $h(u, u_1)$ is constant for $u \in U_i$, $h(u_1, u_1) = 0$. But $u_1 = u_1$, so $h(u_1, u_1)$ corresponds to the case $x = y, x' = y'$, which is undefined. This is not a contradiction yet. However, consider the extension of f. Any protocol for f computes a total function F that extends f. $F((u, 0), (v, 1)) = g(u, v)$ and $F((u, 0), (v, 0)) = h(u, v)$. We established $g(u, u) = 1$ and $h(u, v) = 0$ for $u \neq v$. Consider the specific extension F. The set of inputs where $x' = 0, y' = 1$ forces F to act as Equality on the diagonal. The set of inputs where $x' = 0, y' = 0$ forces F to act as Inequality off the diagonal. Specifically, for $u, v \in U_i$: $g(u, v)$ depends only on v. Let $g(u, v) = G(v)$. $h(u, v)$ depends only on v. Let $h(u, v) = H(v)$. We require $G(u) = 1$ for all $u \in U_i$ (diagonal condition). We require $H(v) = 0$ for all $u \in U_i \setminus \{v\}$ (off-diagonal condition). Now consider the ambiguity. The protocol P is fixed. Does P output 1 or 0 on the undefined inputs? Consider $u_1, u_2 \in U_i$. $g(u_2, u_1) = G(u_1) = 1$. (Input $((u_2, 0), (u_1, 1))$ is undefined). $h(u_1, u_1) = H(u_1) = 0$ (since $h(u_2, u_1) = 0$). (Input $((u_1, 0), (u_1, 0))$ is undefined). So for $v = u_1$, Bob outputs 1 in mode $y' = 1$ and 0 in mode $y' = 0$, regardless of whether $u = u_1$ or $u = u_2$. This means Bob cannot distinguish u_1 from u_2. Specifically, if Bob has $v = u_1$: If Alice has u_1: Run 1 gives 1. Run 2 gives 0. If Alice has u_2: Run 1 gives 1. Run 2 gives 0. Bob sees $(1, 0)$ in both cases. But for f to be computed correctly: If Alice has u_1 ($x = y$): Run 1 is defined ($f = 1$), so output must be 1. Run 2 is undefined. If Alice has u_2 ($x \neq y$): Run 1 is undefined. Run 2 is defined ($f = 0$), so output must be 0. The protocol outputs $(1, 0)$ in both cases. This is consistent with the requirements (1 matches 1, 0 matches 0). Wait, this implies such a protocol *can* exist with low communication? No. Re-evaluate the requirements. f requires $x = y \implies 1$ (when $x' \neq y'$) and $x \neq y \implies 0$ (when $x' = y'$). The protocol on U_i behaves as: On $((u, 0), (v, 1))$: Output $G(v)$. Since $u = v$ must be 1, $G(v) = 1$ for all $v \in U_i$. So

$g(u,v) = 1$ for all $u, v \in U_i$. On $((u,0),(v,0))$: Output $H(v)$. Since $u \neq v$ must be 0, $H(v) = 0$ for all $v \in U_i$ (provided $|U_i| \geq 2$). So $h(u,v) = 0$ for all $u, v \in U_i$. Now check consistency. For $u, v \in U_i$: $P((u,0),(v,1)) = 1$. $P((u,0),(v,0)) = 0$. This protocol satisfies f on the restricted domain. However, this protocol depends on y'. Does it satisfy f globally? Consider inputs where $x' = 1, y' = 0$. $f = 1$ if $x = y$. $f = 0$ if $x \neq y$ (impossible since $x' \neq y'$). So on $x' = 1, y' = 0$, f is defined only if $x = y$, value 1. If $x \neq y$, undefined. The protocol must output 1 if $x = y$. But Alice sends the same message for all $u \in U_i$. Bob must output 1 if $x = y$ and something else if undefined? No constraint. So Bob can just output 1. This suggests $D(f)$ could be small? No. The problem statement asserts $D(f) = \Omega(n)$. The flaw in the pigeonhole argument is assuming Bob's behavior depends *only* on v. Bob's behavior depends on v AND y'. But we fixed y' in the sub-functions. Let's use the rank bound on the matrix M where $M_{uv} = f((u,u),(v,\bar{v}))$. We have $M_{uu} = 1$ and $M_{u\bar{u}} = 0$. Consider the fooling set $S = \{((u,u),(u,\bar{u})) \mid u \in \{0,1\}^{n/2}\}$. For any distinct pairs $((u,u),(u,\bar{u}))$ and $((v,v),(v,\bar{v}))$, the cross term is $((u,u),(v,\bar{v}))$. Here $x = u, y = v \implies x \neq y$. $x' = u, y' = \bar{v}$. If we choose the set of code words $C \subseteq \{0,1\}^{n/2}$ such that for all distinct $u, v \in C$, $u \neq \bar{v}$, then the cross terms have $x' \neq y'$. Then $x \neq y$ and $x' \neq y'$, so f is undefined. This prevents the standard fooling set argument. However, we can use the fact that f contains $NEQ(x', y')$ as a subproblem. Consider the projection of f onto inputs where $x = y$. $f(x, x', x, y') = 1$ if $x' \neq y'$. $f(x, x', x, y')$ is undefined if $x' = y'$ (since $x = y$ contradicts $x \neq y$). So on the slice $x = y$, f is defined only for $x' \neq y'$. Consider the projection onto inputs where $x \neq y$. $f(x, x', y, y') = 0$ if $x' = y'$. Undefined if $x' \neq y'$. A protocol for f must handle both. Consider the function $F(x', y') = 1$ if $x' \neq y'$, 0 if $x' = y'$. We claim $D(f) \geq D(F) = \Omega(n)$. Suppose we have a protocol P for f. Construct a protocol for $F(a,b)$. Alice has a, Bob has b. Alice sets $x' = a$. Bob sets $y' = b$. They need to set x, y. They use public randomness to pick $r \in \{0,1\}^{n/2}$. Alice sets $x = r$. Bob sets $y = r$. They run P on $((r,a),(r,b))$. If $a \neq b$: Input is $x = y, x' \neq y'$. $f = 1$. P outputs 1. If $a = b$: Input is $x = y, x' = y'$. Undefined. This direction fails. Try: Alice sets $x = r$. Bob sets $y = \bar{r}$. If $a = b$: Input $x \neq y, x' = y'$. $f = 0$. P outputs 0. If $a \neq b$: Input $x \neq y, x' \neq y'$. Undefined. Combine: Alice and Bob use shared randomness to pick r. Run 1: $x = r, y = r$. Input $((r,a),(r,b))$. Output o_1. Run 2: $x = r, y = \bar{r}$. Input $((r,a),(\bar{r},b))$. Output o_2. If $a \neq b$: Run 1 is defined (1). $o_1 = 1$. Run 2 is undefined. If $a = b$: Run 1 is undefined. Run 2 is defined (0). $o_2 = 0$. Distinguisher: If $o_1 = 1, o_2 = 1 \implies a \neq b$. If $o_1 = 0, o_2 = 0 \implies a = b$. If $o_1 = 1, o_2 = 0 \implies$ Ambiguous. If $o_1 = 0, o_2 = 1 \implies$ Impossible. Does there exist a protocol P that always outputs $(1,0)$? This would require P to output 1 on $x = y$ (even if $x' = y'$) and 0 on $x \neq y$ (even if $x' \neq y'$). Such a protocol computes $EQ(x,y)$. Thus, any protocol for f must either have high complexity to compute f correctly, OR it effectively computes $EQ(x,y)$ to "cheat" on the undefined inputs. Since $D(EQ(x,y)) = \Omega(n)$, the cost is $\Omega(n)$ in the cheating case. In the non-cheating case, it computes $NEQ(x', y')$? Formally, let P be a protocol for f. Define $P_{diag}(x,y) = P((x,0),(y,0))$. If P is correct for f, then for $x \neq y$, $P_{diag}(x,y) = 0$ (since $x' = y' = 0 \implies f = 0$). If P has cost C, then P_{diag} has cost C. P_{diag} is a protocol that outputs 0 whenever $x \neq y$. On $x = y$, it is unconstrained. To minimize cost, P_{diag} could be the constant 0 function. But we also have $P_{off}(x,y) = P((x,0),(y,1))$. For $x = y$, $P_{off}(x,y) = 1$ (since $x' = 0, y' = 1 \implies f = 1$). On $x \neq y$, unconstrained. P_{off} could be constant 1. But P_{diag} and P_{off} are the *same* protocol P with different Bob inputs. Bob knows if he inputs 0 or 1. Let $T_A(x)$ be Alice's message. Bob computes $Out(T_A(x), y, y')$. We need: 1. $x \neq y \implies Out(T_A(x), y, 0) = 0$. 2. $x = y \implies Out(T_A(x), y, 1) = 1$. Consider the set of inputs $S = \{(w,w) \mid w \in \{0,1\}^{n/2}\}$. For any w, Bob must be able to output 1 if $y = w, y' = 1$. For any $z \neq w$, Bob must be able to output 0 if $y = z, y' = 0$. Suppose Alice sends k bits. She partitions inputs into 2^k sets. Let U be a set in the partition. Let $w_1, w_2 \in U$. Alice sends same T. Bob has input $y = w_1, y' = 1$. He must output 1 (since x could be w_1). Bob has input $y = w_1, y' = 0$. He must output 0 (since x could be $w_2 \neq w_1$). So Bob's output is determined by y'. $Out(T, y, 1) = 1$ and $Out(T, y, 0) = 0$. Now check validity for $x = w_2 \in U$. Bob has $y = w_1, y' = 0$. $x \neq y$. Requirement: Output 0. Bob outputs $Out(T, w_1, 0) = 0$. OK. Bob has $y = w_1, y' = 1$. $x \neq y$. Requirement: None (undefined). Bob outputs $Out(T, w_1, 1) = 1$. OK. This seems to imply constant cost is possible? Wait. Check $x = w_2, y = w_2$. Bob has $y = w_2, y' = 1$. Requirement: Output 1. Bob outputs $Out(T, w_2, 1) = 1$. OK. Bob has $y = w_2, y' = 0$. Requirement: None. Bob outputs $Out(T, w_2, 0) = 0$. OK. This protocol works for these

cases! Where does it fail? It fails if there is a case where $x = y$ requires 0, or $x \neq y$ requires 1. But $f = 1 \implies x = y$. $f = 0 \implies x \neq y$. So $x = y$ never requires 0. $x \neq y$ never requires 1. So the protocol "Output 1 if $y' = 1$, 0 if $y' = 0$" works? Check $f(x, x', y, y')$. If $y' = 1$: Output 1. Correct if $f = 1$. $f = 1 \implies x = y, x' \neq y'$. If $y' = 1$, we need $x' \neq 1$. If $x' = 1$, then $x' = y'$, so $f \neq 1$. So if $f = 1$, then $x' \neq 1$ is guaranteed? No. If $x' = 1, y' = 1$, then $x' = y'$. If $x = y$, then f is undefined. If $x \neq y$, then $f = 0$. In this case ($x \neq y, x' = 1, y' = 1$), the protocol outputs 1. But $f = 0$. Error! So the simple protocol fails when $x' = y' = 1$. It must output 0 when $x' = y'$. So Bob needs to know x'. Alice must send information about x'. To distinguish $x' = 0$ from $x' = 1$ (when $y' = 1$), Alice must send x'. This requires $\Omega(n)$ bits. Specifically, consider $x = y$. $f = 1 \iff x' \neq y'$. If Alice sends nothing, Bob has y, y'. He doesn't know x'. He must output correct value for all x'. If $y' = 0$: If $x' = 1$, $f = 1$. Output 1. If $x' = 0$, undefined. If $y' = 1$: If $x' = 0$, $f = 1$. Output 1. If $x' = 1$, undefined. This doesn't force bits. But consider $x \neq y$. $f = 0 \iff x' = y'$. If $y' = 0$: If $x' = 0$, $f = 0$. Output 0. If $x' = 1$, undefined. If $y' = 1$: If $x' = 1$, $f = 0$. Output 0. If $x' = 0$, undefined. Now combine. Fix $y' = 0$. If $x' = 1$ (and $x = y$), output 1. If $x' = 0$ (and $x \neq y$), output 0. Alice has x, x'. Bob has y. Bob needs to distinguish $(x' = 1, x = y)$ from $(x' = 0, x \neq y)$. Alice sends message $m(x, x')$. Bob computes $O(m, y)$. If $x' = 1$, $O(m, x) = 1$. If $x' = 0$, $O(m, y) = 0$ for all $y \neq x$. This is the "Greater Than" or "Membership" problem structure. Let $S_1 = \{(x, 1)\}$. $S_0 = \{(x, 0)\}$. For any x, Alice $(x, 1)$ wants Bob (x) to output 1. For any x, Alice $(x, 0)$ wants Bob (y) to output 0 for all $y \neq x$. Suppose Alice sends k bits. Let U be a set of inputs (x, x') sharing a message. If U contains $(x, 1)$ and $(z, 0)$, then: Bob on x must output 1 (due to $(x, 1)$). Bob on x must output 0 (due to $(z, 0)$ if $x \neq z$). So if U contains $(x, 1)$ and $(z, 0)$ with $x \neq z$, contradiction. So for each message, the set of x's with $x' = 1$ and the set of x's with $x' = 0$ must be "consistent". Specifically, if $(x, 1) \in U$, then for all $(z, 0) \in U$, we must have $z = x$. So U can contain at most one element with $x' = 0$ if it contains any with $x' = 1$? No. If $(x, 1) \in U$, then Bob (x) outputs 1. If $(z, 0) \in U$, Bob (y) outputs 0 for all $y \neq z$. So Bob (x) outputs 0 if $x \neq z$. Contradiction if $x \neq z$. So if $(x, 1) \in U$ and $(z, 0) \in U$, we must have $x = z$. So U cannot contain $(x, 1)$ and $(z, 0)$ for distinct x, z. Let $A = \{x \mid (x, 1) \in U\}$ and $B = \{z \mid (z, 0) \in U\}$. Then $A \times B \subseteq \{(x, x)\}$. This implies $|A| \cdot |B| \leq 1$ (assuming non-empty). So either A is empty, B is empty, or both are $\{x\}$. So each leaf can cover at most 1 "mixed" pair, or many of one type. But we have $2^{n/2}$ inputs of type $(x, 1)$ and $2^{n/2}$ of type $(x, 0)$. To cover them, we need many leaves. Specifically, consider the set of pairs $P = \{((x, 1), (x, 0)) \mid x \in \{0, 1\}^{n/2}\}$. These pairs cannot share a leaf. Thus we need at least $2^{n/2}$ leaves. $D(f) \geq n/2$. □

B.13 Problem 1.13

B.13.1 Problem Statement

Lemma B.13. *For a Boolean function g, define $g^{\wedge k}$ by*

$$g^{\wedge k}(x_1, x_2, \ldots, x_k, y_1, \ldots, y_k) = \bigwedge_{i=1}^{k} g(x_i, y_i).$$

If $g^{\wedge k}$ has a 1-cover of size 2^ℓ, then g has a 1-cover of size $2^{\ell/k}$.

B.13.2 GPT

Proof. Let $g^{\wedge k}$ be the Boolean function defined by

$$g^{\wedge k}(x_1, x_2, \ldots, x_k, y_1, \ldots, y_k) = \bigwedge_{i=1}^{k} g(x_i, y_i).$$

Suppose that $g^{\wedge k}$ has a 1-cover of size 2^ℓ. This means there exists a set $C \subseteq \{0, 1\}^{2k}$ with $|C| = 2^\ell$ such that for all inputs $(x_1, x_2, \ldots, x_k, y_1, \ldots, y_k) \in \{0, 1\}^{2k}$, we have

$$g^{\wedge k}(x_1, x_2, \ldots, x_k, y_1, \ldots, y_k) = 1 \quad \text{if and only if} \quad (x_1, x_2, \ldots, x_k, y_1, \ldots, y_k) \in C.$$

Now, consider the Boolean function g, and let us investigate its 1-cover. We aim to show that g has a 1-cover of size $2^{\ell/k}$.

For each $i \in \{1, 2, \ldots, k\}$, define the projection of the set C onto the i-th coordinate by

$$C_i = \{(x_i, y_i) \mid (x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_k, y_1, \ldots, y_{i-1}, y_i, y_{i+1}, \ldots, y_k) \in C\}.$$

This projection corresponds to the set of pairs (x_i, y_i) for which the function $g(x_i, y_i) = 1$ whenever the remaining coordinates $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k, y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_k)$ are chosen appropriately.

We observe that since C is a set of size 2^ℓ, and each projection C_i involves fixing all the coordinates except for (x_i, y_i), the size of each C_i is at most $2^{\ell/k}$, since the total number of coordinates is $2k$, and there are k coordinates of the form (x_i, y_i).

Thus, each projection C_i has at most $2^{\ell/k}$ elements. Since the set C is a product of k such projections, the total number of distinct pairs $(x_1, x_2, \ldots, x_k, y_1, \ldots, y_k)$ that satisfy $g(x_1, x_2, \ldots, x_k, y_1, \ldots, y_k) = 1$ is at most $2^{\ell/k}$.

Therefore, the Boolean function g has a 1-cover of size at most $2^{\ell/k}$, completing the proof. □

B.13.3 Claude

Proof. Let C be a 1-cover of $g^{\wedge k}$ of size 2^ℓ. We will construct a 1-cover of g of size $2^{\ell/k}$.

Since C is a 1-cover of $g^{\wedge k}$, for every input $(x_1, x_2, \ldots, x_k, y_1, \ldots, y_k)$ such that $g^{\wedge k}(x_1, x_2, \ldots, x_k, y_1, \ldots, y_k) = 1$, there exists a rectangle $R \in C$ that covers this input and is contained in the 1-inputs of $g^{\wedge k}$.

Each rectangle $R \in C$ can be written as $R = A \times B$ where $A \subseteq \{0, 1\}^{2k}$ is a set of x-coordinates and $B \subseteq \{0, 1\}^{2k}$ is a set of y-coordinates (assuming g is a function of two variables).

By the definition of $g^{\wedge k}$, we have $g^{\wedge k}(x_1, \ldots, x_k, y_1, \ldots, y_k) = 1$ if and only if $g(x_i, y_i) = 1$ for all $i \in [k]$.

For any rectangle $R = A \times B$ in the 1-cover of $g^{\wedge k}$, if $(x_1, \ldots, x_k) \in A$ and $(y_1, \ldots, y_k) \in B$, then $g(x_i, y_i) = 1$ for all $i \in [k]$.

This means that for each $i \in [k]$, the projection of R onto the i-th coordinate pair gives a rectangle R_i in the domain of g that is contained in the 1-inputs of g.

Now, consider the collection $\mathcal{C}$ of all rectangles obtained by taking projections of rectangles in C onto any single coordinate pair. More precisely, for each $R \in C$ and each $i \in [k]$, let R_i denote the projection of R onto the i-th coordinate pair.

The key observation is that if (x, y) is a 1-input of g, then the input $(x, \ldots, x, y, \ldots, y)$ (where x appears k times and y appears k times) is a 1-input of $g^{\wedge k}$. Therefore, there exists some rectangle $R \in C$ covering this input, and the projection R_i for any $i \in [k]$ covers (x, y).

Thus, the collection of all projections forms a 1-cover of g. However, this collection may have size up to $k \cdot 2^\ell$.

To obtain a 1-cover of size $2^{\ell/k}$, we use the following argument: Since C has size 2^ℓ, and each rectangle in C projects to k rectangles (one for each coordinate pair), we can select a subset of these projections that covers all 1-inputs of g.

By the pigeonhole principle and properties of covers, there exists a coordinate $i \in [k]$ such that the projections $\{R_i : R \in C\}$ can be refined to a 1-cover of g of size at most $2^\ell/k$. Since the size of a cover must be a power of 2 in the context of communication complexity, we obtain a 1-cover of size $2^{\ell/k}$.

Alternatively, by a more careful analysis using the fact that the 1-cover complexity is subadditive under the AND operation, we have that if $g^{\wedge k}$ has 1-cover complexity ℓ, then g has 1-cover complexity at most ℓ/k, which means g has a 1-cover of size $2^{\ell/k}$. □

B.13.4 Gemini

Proof. Let X and Y denote the domains of the inputs x and y for the function g. We associate g with its communication matrix M_g of dimension $|X| \times |Y|$, defined by $(M_g)_{x,y} = g(x, y)$. A 1-cover of g corresponds to a set of monochromatic 1-rectangles (submatrices of all ones) that cover the 1-entries of M_g. The size of a minimal 1-cover of g is equal to the Boolean rank of M_g, denoted by $\mathrm{rk}_B(M_g)$.

The function $g^{\wedge k}$ takes inputs from $X^k \times Y^k$. For inputs $\vec{x} = (x_1, \ldots, x_k)$ and $\vec{y} = (y_1, \ldots, y_k)$, the function is defined by

$$g^{\wedge k}(\vec{x}, \vec{y}) = \bigwedge_{i=1}^{k} g(x_i, y_i).$$

This implies that $g^{\wedge k}(\vec{x}, \vec{y}) = 1$ if and only if $g(x_i, y_i) = 1$ for all $i \in \{1, \ldots, k\}$. In terms of matrices, the communication matrix of $g^{\wedge k}$ is the k-fold Kronecker product of M_g, denoted by $M_g^{\otimes k}$. Consequently, the size of the minimal 1-cover of $g^{\wedge k}$ is given by $\mathrm{rk}_B(M_g^{\otimes k})$.

We are given that $g^{\wedge k}$ has a 1-cover of size 2^ℓ. Therefore,

$$\mathrm{rk}_B(M_g^{\otimes k}) \leq 2^\ell.$$

A known property of the Boolean rank is its behavior under the Kronecker product. Specifically, for any Boolean matrix A, the Boolean rank satisfies

$$\mathrm{rk}_B(A^{\otimes k}) = (\mathrm{rk}_B(A))^k.$$

Applying this property to M_g, we have

$$(\mathrm{rk}_B(M_g))^k = \mathrm{rk}_B(M_g^{\otimes k}) \leq 2^\ell.$$

Taking the k-th root of both sides yields

$$\mathrm{rk}_B(M_g) \leq (2^\ell)^{1/k} = 2^{\ell/k}.$$

Since $\mathrm{rk}_B(M_g)$ represents the size of the minimal 1-cover of g, it follows that g has a 1-cover of size at most $2^{\ell/k}$. □

B.14 Problem 1.14

B.14.1 Problem Statement

Theorem B.14. *If $g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ requires c bits of communication, then any protocol computing both $\bigwedge_{i=1}^k g(x_i, y_i)$ and $\bigvee_{i=1}^k g(x_i, y_i)$ requires $k(\sqrt{c/2} - \log n - 1)$ bits of communication.*

B.14.2 GPT

Proof. Fix a deterministic protocol model for communication complexity, and let $D(g) = c$ denote the deterministic communication complexity of g. For $k \geq 1$, define the two functions on k independent instances

$$G_\wedge((x_1, \ldots, x_k), (y_1, \ldots, y_k)) = \bigwedge_{i=1}^{k} g(x_i, y_i), \qquad G_\vee((x_1, \ldots, x_k), (y_1, \ldots, y_k)) = \bigvee_{i=1}^{k} g(x_i, y_i).$$

A protocol computing both outputs is a deterministic protocol whose transcript determines the ordered pair $(G_\wedge, G_\vee) \in \{(0,0), (0,1), (1,1)\}$.

Let Π be any deterministic protocol that, on input $(x_1, \ldots, x_k)$ to Alice and $(y_1, \ldots, y_k)$ to Bob, outputs both $G_\wedge$ and $G_\vee$ and communicates C bits. We prove the contrapositive: if $C < k(\sqrt{c/2} - \log n - 1)$ then there exists a deterministic protocol for g that communicates fewer than c bits, contradicting $D(g) = c$.

Set

$$a = \frac{C}{k}.$$

Assume $a < \sqrt{c/2} - \log n - 1$, equivalently

$$a + \log n + 1 < \sqrt{c/2}.$$

Define an integer

$$t = \lceil a + \log n + 1 \rceil,$$

so that $t < \sqrt{c/2} + 1$ and in particular $2t^2 \le c$ holds whenever $t \le \sqrt{c/2}$; since $t \le a + \log n + 2 < \sqrt{c/2} + 1$, we may and do replace t by $\min\{t, \lfloor \sqrt{c/2} \rfloor\}$ so that

$$t \le a + \log n + 1 \qquad \text{and} \qquad 2t^2 \le c.$$

We now construct a deterministic protocol $\Pi^\star$ for g with communication at most $2t^2$, yielding $D(g) \le 2t^2 \le c$, and in fact $D(g) \le 2t^2 < c$ because $t \le \lfloor \sqrt{c/2} \rfloor$ implies $2t^2 \le c - 1$ unless c is exactly twice a square; in the remaining equality case, we reduce t by 1 and still have $t \le a + \log n + 1$ because $a + \log n + 1 < \sqrt{c/2} = \sqrt{t^2}$ forces $a + \log n + 1 \le t - 1$.

The construction relies on the following deterministic embedding lemma.

Let $U = \{0,1\}^n$. For each $u \in U$, fix an arbitrary ordering of U and let $\mathrm{ind}(u) \in \{0,1\}^{\lceil \log |U| \rceil} = \{0,1\}^n$ be the binary encoding of its index in this ordering. Note that ind is computable locally by both players without communication once u is known.

Define, for any pair $(x, y) \in U \times U$ and any pair of t-bit strings $(p, q) \in \{0,1\}^t \times \{0,1\}^t$, the derived strings

$$\phi_A(x, p) \in U, \qquad \phi_B(y, q) \in U,$$

as follows. Let $p = p_1 \cdots p_t$ and $q = q_1 \cdots q_t$. Consider the map $\sigma : \{0,1\}^t \to U$ that maps a t-bit string to the first t bits of an element of U and pads the remaining $n - t$ bits with zeros; this is well-defined since $t \le n$ (otherwise replace $\log n$ by n in what follows; since $\log n \le n$, this only strengthens the lower bound). Set

$$\phi_A(x, p) = x \oplus \sigma(p), \qquad \phi_B(y, q) = y \oplus \sigma(q),$$

where $\oplus$ is bitwise XOR in U.

For each $j \in \{1, \ldots, k\}$ and each t-bit string $r \in \{0,1\}^t$, define a k-tuple input to Π by

$$x_i^{(j,r)} = \begin{cases} \phi_A(x, r) & i = j, \\ \phi_A(0^n, r) & i \ne j, \end{cases} \qquad y_i^{(j,r)} = \begin{cases} \phi_B(y, r) & i = j, \\ \phi_B(0^n, r) & i \ne j, \end{cases}$$

where 0^n is the all-zero string.

Let the outputs of Π on this k-tuple be denoted

$$\Pi_\wedge(j, r) \in \{0,1\}, \qquad \Pi_\vee(j, r) \in \{0,1\}.$$

Since all coordinates $i \ne j$ are equal to the fixed pair $(\phi_A(0^n, r), \phi_B(0^n, r))$, their g-value depends only on r; define

$$b(r) = g(\phi_A(0^n, r), \phi_B(0^n, r)) \in \{0,1\}.$$

Then, by definition of $G_\wedge$ and $G_\vee$,

$$\Pi_\wedge(j, r) = b(r)^{k-1} \wedge g(\phi_A(x, r), \phi_B(y, r)), \qquad \Pi_\vee(j, r) = b(r)^{k-1} \vee g(\phi_A(x, r), \phi_B(y, r)),$$

where $b(r)^{k-1}$ denotes the AND (respectively OR) of $k - 1$ copies of $b(r)$, so $b(r)^{k-1} = b(r)$ for both AND and OR because $b(r) \in \{0,1\}$. Hence

$$\Pi_\wedge(j, r) = b(r) \wedge g(\phi_A(x, r), \phi_B(y, r)), \qquad \Pi_\vee(j, r) = b(r) \vee g(\phi_A(x, r), \phi_B(y, r)).$$

Therefore, the pair $(\Pi_\wedge(j, r), \Pi_\vee(j, r))$ determines $g(\phi_A(x, r), \phi_B(y, r))$ uniquely whenever $b(r) = 0$ or $b(r) = 1$ as follows:

$$b(r) = 0 \implies (\Pi_\wedge(j, r), \Pi_\vee(j, r)) = (0, g(\phi_A(x, r), \phi_B(y, r))),$$

$$b(r) = 1 \implies (\Pi_\wedge(j, r), \Pi_\vee(j, r)) = (g(\phi_A(x, r), \phi_B(y, r)), 1).$$

In either case, $g(\phi_A(x, r), \phi_B(y, r))$ is computable from $b(r)$ and the two output bits of Π without further communication. Since $b(r)$ depends only on r and the fixed function g, both players can compute $b(r)$ locally once r is agreed upon.

We now show how to compute $g(x, y)$ itself. Consider the set of t-bit masks $r \in \{0,1\}^t$. For each such r, define the derived instance

$$(x_r, y_r) = (\phi_A(x, r), \phi_B(y, r)).$$

Because ϕ_A and ϕ_B are XOR-masks on the first t coordinates, the mapping $(x, y) \mapsto (x_r, y_r)$ is a bijection on $U \times U$ for each fixed r, and in particular the multiset $\{(x_r, y_r) : r \in \{0,1\}^t\}$ uniquely determines (x, y) and vice versa. Moreover, for each r,

$$x = x_r \oplus \sigma(r), \qquad y = y_r \oplus \sigma(r).$$

Define a $t \times t$ grid of masks by $r^{(s)}$ for $s \in \{0,1\}^t$, and consider the t-bit Walsh-Hadamard character

$$\chi_s(r) = (-1)^{\langle s, r\rangle},$$

where $\langle s, r\rangle$ is the inner product modulo 2. Define the signed sum

$$S(x, y) = \sum_{r \in \{0,1\}^t} \chi_s(r) \cdot (-1)^{g(x_r, y_r)}.$$

For each fixed (x, y), the value of $S(x, y)$ depends on the full truth table of the function $r \mapsto g(x_r, y_r)$ over $\{0,1\}^t$. Since ϕ_A, ϕ_B are bijections in (x, y) for each r, the mapping $(x, y) \mapsto (r \mapsto g(x_r, y_r))$ ranges over at most 2^{2n} possible Boolean functions on $\{0,1\}^t$. Hence there exists some $s \in \{0,1\}^t$ such that the sign of $S(x, y)$ determines $g(x, y)$ for all (x, y), because the family of signs of the 2^t Walsh-Hadamard transforms separates all Boolean functions on $\{0,1\}^t$ and thus separates the at most 2^{2n} functions realized by (x, y) once $t \geq \log(2n) + 1$; indeed, $t \geq \log n + 1$ implies $2^t > 2^{\log n+1} = 2n$, and thus there are enough characters to shatter the class of size at most 2^{2n}. Fix such an s once and for all, as advice depending only on g and n.

Given this fixed s, it suffices for Alice and Bob to compute $g(x_r, y_r)$ for all $r \in \{0,1\}^t$ and then compute $\mathrm{sign}(S(x, y))$ locally. We now compute all these values using Π with total communication at most $2t^2$.

To do so, Alice and Bob first agree (without communication) on the list of all $r \in \{0,1\}^t$ in lexicographic order. They will simulate Π on k-tuples in which, for a selected coordinate $j(r) \in \{1, \ldots, k\}$ assigned injectively to each r (possible since $k \geq 2^t$ will be ensured below), the $j(r)$-th instance is (x_r, y_r) and all other instances are fixed to $(\phi_A(0^n, r), \phi_B(0^n, r))$ depending on that same r. Specifically, for each r, they run Π on the input $(x_i^{(j(r),r)})_{i=1}^k, (y_i^{(j(r),r)})_{i=1}^k$ and obtain $(\Pi_\wedge(j(r), r), \Pi_\vee(j(r), r))$, from which they reconstruct $g(x_r, y_r)$ as shown above.

Each such run uses C bits of communication, so doing this for all 2^t masks would cost $2^t C$, which is too large. The key point is that Π is a deterministic protocol whose transcript is a function of the full k-tuple inputs; if we restrict to a subset of inputs where all but one coordinate are fixed in a structured way, then the number of distinct transcripts is bounded by 2^C, and by an averaging argument there exists a coordinate $j^\star$ such that, when only the $j^\star$-th coordinate varies among 2^t possibilities, the average transcript length per possibility is at most $C/k = a$.

Formally, for each transcript τ of Π, let R_τ be the corresponding monochromatic rectangle in $U^k \times U^k$ on which Π outputs a fixed pair. Fix any $r \in \{0,1\}^t$ and consider the set

$$\mathcal{I}_r = \{((x_i^{(j,r)})_{i=1}^k, (y_i^{(j,r)})_{i=1}^k) : j \in \{1, \ldots, k\}\}.$$

The sets $\mathcal{I}_r$ for different r are disjoint. The protocol partitions each $\mathcal{I}_r$ into at most 2^C rectangles. Hence, for each r, the number of distinct transcripts appearing on $\mathcal{I}_r$ is at most 2^C, and therefore there exists a transcript τ_r that occurs on at least $k/2^C$ values of j in $\mathcal{I}_r$. Let

$$J_r = \{j \in \{1, \ldots, k\} : \Pi \text{ has transcript } \tau_r \text{ on input indexed by } j \text{ in } \mathcal{I}_r\},$$

so $|J_r| \geq k/2^C$. Since $2^C = (2^a)^k$, we have $k/2^C \geq 1$ and thus each J_r is nonempty. Choose $j(r) \in J_r$ deterministically as the least element.

Now Alice and Bob can compute $j(r)$ without communication because $j(r)$ depends on (x, y) through the transcript, but we will instead communicate the index $j(r)$ explicitly. Since $j(r) \in \{1, \ldots, k\}$, it costs at most $\lceil \log k \rceil$ bits to specify. We ensure $k \leq 2^n$ so $\log k \leq n$, and since $t \leq \log n + a + 1 \leq n$, we have $\log k \leq t + \log n$; we will absorb this into the t overhead.

For each $r \in \{0, 1\}^t$, Alice sends $j(r)$ to Bob using at most t bits, because we require $k \leq 2^t$ by choosing an injective assignment of masks to indices and, if necessary, padding the protocol to $k' = 2^t$ coordinates by duplicating coordinates; this can only reduce the per-coordinate cost lower bound and does not increase the required communication for computing both $G_\wedge$ and $G_\vee$ on the original k coordinates. Thus, we may assume $k = 2^t$ and hence $j(r)$ can be encoded in exactly t bits.

After receiving $j(r)$, both parties simulate Π on the k-tuple input determined by $(j(r), r)$ as above. Because $j(r) \in J_r$, the transcript of Π on this restricted input is τ_r, which is fixed given r and independent of (x, y) except through the $j(r)$ choice already communicated. Therefore, in this simulation, the only communication that actually occurs is bounded by the length of τ_r, and summing over all r yields total simulated communication at most

$$\sum_{r \in \{0,1\}^t} |\tau_r| \leq 2^t \cdot a \cdot k/2^t = ak,$$

and since $k = 2^t$ this equals $a2^t$. We instead use the deterministic bound that the average transcript length over the k choices of j in $\mathcal{I}_r$ is at most $C/k = a$, hence the least frequent transcript length among those appearing at frequency at least 1 is at most a, and thus $|\tau_r| \leq a$ for all r by the choice of τ_r minimizing length among those with maximal frequency; this can be ensured by tie-breaking. Consequently each simulation costs at most a bits.

Thus, for each r, the communication consists of sending $j(r)$ (at most t bits) plus simulating Π with transcript τ_r (at most a bits). The total communication to compute all $g(x_r, y_r)$ for $r \in \{0, 1\}^t$ is at most

$$2^t(t + a).$$

Finally, since $t \leq a + \log n + 1$, we have $t + a \leq 2t$, and with $k = 2^t$ the above becomes $2^t(t + a) \leq 2^t \cdot 2t = 2kt$. Dividing by k yields per-instance cost at most $2t$. Applying this procedure recursively on t levels by splitting the t-bit masks into two halves of size $t/2$ and reusing the same argument on each half yields a protocol for g of cost at most $2t^2$. Concretely, view r as $(r^{(1)}, r^{(2)})$ with $r^{(1)}, r^{(2)} \in \{0, 1\}^{t'}$ where $t' = \lceil t/2 \rceil$, compute $g(x_{r^{(1)}}, y_{r^{(1)}})$ for all $r^{(1)}$ using cost at most $2t'(t' + a) \leq 2t^2/2 = t^2$, and similarly for the second half, giving total at most $2t^2$.

Therefore there exists a deterministic protocol $\Pi^\star$ for g with communication at most $2t^2$. By our choice of t we have $2t^2 < c$, contradicting $D(g) = c$. Hence our assumption $C < k(\sqrt{c/2} - \log n - 1)$ must be false, and every protocol computing both $G_\wedge$ and $G_\vee$ requires at least $k(\sqrt{c/2} - \log n - 1)$ bits of communication. □

B.14.3 Claude

Proof. Let $g : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}$ be a function requiring c bits of communication in the worst case. We prove that any protocol computing both $\bigwedge_{i=1}^k g(x_i, y_i)$ and $\bigvee_{i=1}^k g(x_i, y_i)$ requires $k(\sqrt{c/2} - \log n - 1)$ bits of communication.

Consider a protocol Π that computes both functions. We construct a protocol for g using Π as a subroutine. The key observation is that if we can compute both the AND and OR of k instances of g, we can extract information about individual instances.

For any protocol computing a function $f : X \times Y \to Z$, let $\mathrm{CC}(f)$ denote its communication complexity. By assumption, $\mathrm{CC}(g) = c$.

Consider the following reduction: Given a single instance (x, y) of g, we embed it among k instances. Specifically, we construct inputs $(x_1, y_1), \ldots, (x_k, y_k)$ where one coordinate contains (x, y) and the remaining coordinates are chosen strategically.

Let $\mathcal{R}$ be a combinatorial rectangle in the communication matrix of g that is monochromatic. By the rectangle covering bound, the number of monochromatic rectangles needed to cover all inputs where g evaluates to a constant is at least 2^c.

For the protocol Π computing both $\bigwedge_{i=1}^{k} g(x_i, y_i)$ and $\bigvee_{i=1}^{k} g(x_i, y_i)$, consider the partition of the input space induced by the protocol's communication pattern. Each leaf of the protocol tree corresponds to a combinatorial rectangle in the product space $(\{0,1\}^n)^k \times (\{0,1\}^n)^k$.

By the information-theoretic argument, if Π uses C bits of communication, then it partitions the input space into at most 2^C rectangles. Each such rectangle in the product space projects down to rectangles in each coordinate.

To extract information about a single coordinate, we use the fact that knowing both $\bigwedge_{i=1}^{k} g(x_i, y_i)$ and $\bigvee_{i=1}^{k} g(x_i, y_i)$ allows us to distinguish certain patterns. Specifically, if the AND is 1, then all coordinates must be 1. If the OR is 0, then all coordinates must be 0. The gap between these cases provides information.

By a counting argument, each rectangle in the k-fold product space can be associated with at most n^k distinct configurations when projected to individual coordinates. The number of such rectangles must be at least 2^{kc} to distinguish all possible patterns of g values across k coordinates.

Therefore, we have $2^C \cdot n^k \geq 2^{kc}$, which gives us $2^C \geq 2^{kc}/n^k = 2^{kc - k\log n}$.

Taking logarithms, we obtain $C \geq kc - k\log n$.

However, this bound is too weak. To obtain the stronger bound, we use a more refined information-theoretic argument based on the direct sum theorem for communication complexity. By the information cost method, computing k independent instances requires communication proportional to k times the information cost of a single instance.

The information cost of g is at least $\sqrt{c/2}$ bits by standard results in information complexity theory. Accounting for the correlation between AND and OR computations and the overhead of $\log n + 1$ bits per instance for coordination, we obtain:

$$C \geq k(\sqrt{c/2} - \log n - 1).$$

This completes the proof. □

B.14.4 Gemini

Proof. Let P be a deterministic protocol that computes both $F_{AND}(X,Y) = \bigwedge_{i=1}^{k} g(x_i, y_i)$ and $F_{OR}(X,Y) = \bigvee_{i=1}^{k} g(x_i, y_i)$ for inputs $X = (x_1, \ldots, x_k)$ and $Y = (y_1, \ldots, y_k)$, where $x_i, y_i \in \{0,1\}^n$. Let C denote the communication complexity of P. We aim to show that $C \geq k(\sqrt{c/2} - \log n - 1)$, where $c = D(g)$.

We first relate C to the nondeterministic communication complexities of g. Let $N^1(g)$ denote the nondeterministic communication complexity of g (the logarithm of the minimum number of monochromatic 1-rectangles needed to cover the 1-inputs of g), and let $N^0(g)$ denote the nondeterministic communication complexity of the complement of g (covering the 0-inputs).

Step 1: Bounding $N^1(g)$ and $N^0(g)$. Consider the function F_{AND}. The set of inputs where $F_{AND}(X,Y) = 1$ is exactly the set where $g(x_i, y_i) = 1$ for all $i = 1, \ldots, k$. This is the product set $(g^{-1}(1))^k$. The protocol P partitions the input space $\{0,1\}^{nk} \times \{0,1\}^{nk}$ into 2^C combinatorial rectangles. Let $\mathcal{R}_{out}$ be the set of rectangles in the transcript of P where the output for F_{AND} is 1. Since P is correct, the union of these rectangles covers exactly $(g^{-1}(1))^k$. It is a known result in communication complexity that $N^1(g^k) = kN^1(g)$. Since P provides a cover of size at most 2^C for the 1-inputs of g^k, we have:

$$N^1(g^k) \leq C \implies kN^1(g) \leq C \implies N^1(g) \leq \frac{C}{k}.$$

Similarly, consider F_{OR}. The set of inputs where $F_{OR}(X,Y) = 0$ is exactly $(g^{-1}(0))^k$. The rectangles in P where the output for F_{OR} is 0 form a cover for the 0-inputs of g^k. Thus:

$$N^0(g^k) \leq C \implies kN^0(g) \leq C \implies N^0(g) \leq \frac{C}{k}.$$

Step 2: Relating $D(g)$ to $N^0(g)$ and $N^1(g)$. We utilize the standard upper bound relating deterministic complexity to nondeterministic complexity. A known variant of the Aho-Ullman-Yannakakis theorem

states that for any function $g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$:

$$D(g) \leq 2(N^0(g) + \log n + 1)(N^1(g) + \log n + 1).$$

(Note: The standard bound is often cited as $O(N^0(g)N^1(g))$. The specific lower order terms $\log n + 1$ account for the overhead in the protocol construction, specifically for sending indices or handling small cover sizes).

Step 3: Deriving the Lower Bound. We are given that $D(g) = c$. Substituting the bounds from Step 1 into the inequality from Step 2:

$$c \leq 2\left(\frac{C}{k} + \log n + 1\right)\left(\frac{C}{k} + \log n + 1\right).$$

Simplifying the expression:

$$c \leq 2\left(\frac{C}{k} + \log n + 1\right)^2.$$

Dividing by 2 and taking the square root of both sides:

$$\sqrt{\frac{c}{2}} \leq \frac{C}{k} + \log n + 1.$$

Rearranging to solve for C:

$$\frac{C}{k} \geq \sqrt{\frac{c}{2}} - \log n - 1.$$

Multiplying by k:

$$C \geq k\left(\sqrt{\frac{c}{2}} - \log n - 1\right).$$

This concludes the proof. □

C Problems in Chapter 2

C.1 Problem 2.1

C.1.1 Problem Statement

Theorem C.1. *The communication complexity of M is at most* $\operatorname{rank}(M) + 1$.

C.1.2 GPT

Proof. Let M be a communication matrix. We are tasked with proving that the communication complexity of M is at most $\operatorname{rank}(M) + 1$.

We begin by recalling the definition of communication complexity. The communication complexity of a matrix M refers to the minimum number of bits that need to be exchanged between two parties in order to compute the entry of M corresponding to a particular input. This is equivalent to the minimum number of bits required for two parties to compute a function based on the matrix entries, where one party knows the row index and the other knows the column index.

To prove the theorem, we consider the communication protocol that computes the entry $M_{i,j}$ of the matrix for any row i and column j. We assume that the communication complexity is minimized, meaning that we seek the most efficient protocol.

Let us recall that the rank of the matrix M is defined as the dimension of the column space (or equivalently, the row space) of M. The key idea behind the proof is that the communication complexity of M is related to the amount of information that needs to be conveyed to determine the entry $M_{i,j}$.

Consider a communication protocol where the row index i is held by one party (the "sender") and the column index j is held by the other party (the "receiver"). The sender can encode the row i of the matrix M as a vector in $\mathbb{R}^n$ (or equivalently, as a vector in a finite-dimensional vector space) where n is the

number of columns of M. Similarly, the receiver can encode the column j as a vector in $\mathbb{R}^m$, where m is the number of rows of M.

In order to determine the entry $M_{i,j}$, the receiver must obtain enough information from the sender to recover the relevant row of M, or equivalently, the relevant vector in $\mathbb{R}^n$. Since the rank of M is the dimension of the row space, it follows that the receiver needs to know at most $\mathrm{rank}(M)$ independent bits of information in order to recover the row corresponding to the row index i.

Now, we consider the communication between the two parties. The sender can send at most $\mathrm{rank}(M)$ bits of information to the receiver, as this is the minimum number of bits needed to fully recover the row corresponding to index i. Additionally, the receiver must send a single bit of information to indicate the specific entry of the matrix $M_{i,j}$ that corresponds to the given column index j. This extra bit accounts for the fact that the receiver must disambiguate between the different entries in the row.

Thus, the total communication complexity consists of $\mathrm{rank}(M)$ bits to communicate the row information and 1 additional bit to select the correct entry within the row. Therefore, the total communication complexity is at most $\mathrm{rank}(M) + 1$.

This completes the proof. □

C.1.3 Claude

Proof. Let M be an $n \times m$ matrix over some field, and let $r = \mathrm{rank}(M)$. We will show that the deterministic communication complexity of M is at most $r + 1$.

Since $\mathrm{rank}(M) = r$, there exists a rank factorization $M = AB^T$ where A is an $n \times r$ matrix and B is an $m \times r$ matrix. This means that for any row index $x \in [n]$ and column index $y \in [m]$, we have

$$M(x, y) = \sum_{i=1}^{r} A(x, i) \cdot B(y, i) = A_x \cdot B_y$$

where A_x denotes the x-th row of A and B_y denotes the y-th row of B.

We now describe a communication protocol for computing $M(x, y)$ where Alice holds input x and Bob holds input y.

Alice computes the row vector $A_x \in \mathbb{F}^r$ and sends it to Bob. Since A_x consists of r field elements, this requires r units of communication (or $O(r \log |\mathbb{F}|)$ bits if we count bit complexity).

Bob receives A_x, computes the row vector $B_y \in \mathbb{F}^r$, and then computes the inner product

$$M(x, y) = A_x \cdot B_y = \sum_{i=1}^{r} A(x, i) \cdot B(y, i)$$

Bob then sends the result $M(x, y)$ back to Alice, which requires 1 additional unit of communication.

The total communication is $r + 1$ units. Therefore, the communication complexity of M is at most $\mathrm{rank}(M) + 1$.

Alternatively, in the standard model where we count the number of bits communicated, Alice sends r field elements (requiring $O(r \log |\mathbb{F}|)$ bits) and Bob sends one field element (requiring $O(\log |\mathbb{F}|)$ bits), giving total communication $O((r + 1) \log |\mathbb{F}|)$ bits, which is $O(\mathrm{rank}(M) + 1)$ in the appropriate complexity measure. □

C.1.4 Gemini

Proof. Let $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ be the function associated with the communication matrix M, where rows are indexed by $x \in \mathcal{X}$ and columns by $y \in \mathcal{Y}$. We consider the rank of M over the field $\mathbb{F}_2$. Let $r = \mathrm{rank}(M)$.

By the definition of matrix rank over a field, there exists a decomposition of M into the product of two matrices A and B such that $M = AB$, where A is an $|\mathcal{X}| \times r$ matrix and B is an $r \times |\mathcal{Y}|$ matrix, both with entries in $\mathbb{F}_2$.

For any input pair (x, y), the value of the function is given by the entry $M_{x,y}$. Using the decomposition, we can express this entry as the inner product of the x-th row of A and the y-th column of B:

$$M_{x,y} = \sum_{k=1}^{r} A_{x,k} B_{k,y} \pmod 2.$$

Let $u_x = (A_{x,1}, \ldots, A_{x,r}) \in \mathbb{F}_2^r$ be the row vector associated with Alice's input x, and let $v_y = (B_{1,y}, \ldots, B_{r,y}) \in \mathbb{F}_2^r$ be the column vector associated with Bob's input y. Note that Alice can compute u_x solely from x and M, and Bob can compute v_y solely from y and M.

We construct a deterministic communication protocol Π as follows: 1. Alice computes the vector u_x. 2. Alice sends the r bits of u_x to Bob. 3. Bob receives u_x and computes the inner product $\langle u_x, v_y \rangle = \sum_{k=1}^{r} A_{x,k} B_{k,y} \pmod 2$. 4. Since $M_{x,y} = \langle u_x, v_y \rangle$, Bob now knows the value of $f(x, y)$. 5. Bob sends the single bit result $f(x, y)$ to Alice.

The total number of bits communicated in this protocol is r (from Alice to Bob) plus 1 (from Bob to Alice), summing to $r + 1$. Since this protocol computes $f(x, y)$ correctly for all inputs, the deterministic communication complexity $D(M)$ is bounded by the cost of this protocol.

Therefore, we conclude:

$$D(M) \leq r + 1 = \text{rank}(M) + 1.$$

□

C.2 Problem 2.2

C.2.1 Problem Statement

Lemma C.2. *If* $\text{rank}(R) \leq 1$,

$$\text{rank}\left(\begin{bmatrix} R \\ B \end{bmatrix}\right) + \text{rank}\left(\begin{bmatrix} R & A \end{bmatrix}\right) \leq \text{rank}\left(\begin{bmatrix} R & A \\ B & C \end{bmatrix}\right) + 3.$$

C.2.2 GPT

Proof. Let R, A, B, and C be matrices with appropriate dimensions, and assume that $\text{rank}(R) \leq 1$. We are tasked with proving the inequality

$$\text{rank}\left(\begin{bmatrix} R & B \end{bmatrix}\right) + \text{rank}\left(\begin{bmatrix} R & A \end{bmatrix}\right) \leq \text{rank}\left(\begin{bmatrix} R & A & C \end{bmatrix}\right) + 3.$$

Step 1: Rank of Block Matrices

By the rank subadditivity property of block matrices, we know that for any matrices X and Y with compatible dimensions:

$$\text{rank}\left(\begin{bmatrix} X & Y \end{bmatrix}\right) \leq \text{rank}(X) + \text{rank}(Y).$$

Thus, we apply this property to the matrices involved in the statement of the theorem.

- For $\begin{bmatrix} R & B \end{bmatrix}$, we have:

$$\text{rank}\left(\begin{bmatrix} R & B \end{bmatrix}\right) \leq \text{rank}(R) + \text{rank}(B).$$

- For $\begin{bmatrix} R & A \end{bmatrix}$, we have:

$$\text{rank}\left(\begin{bmatrix} R & A \end{bmatrix}\right) \leq \text{rank}(R) + \text{rank}(A).$$

- For $\begin{bmatrix} R & A & C \end{bmatrix}$, we have:

$$\text{rank}\left(\begin{bmatrix} R & A & C \end{bmatrix}\right) \leq \text{rank}(R) + \text{rank}(A) + \text{rank}(C).$$

Step 2: Using the Assumption on rank(R)

We are given that $\text{rank}(R) \leq 1$. Therefore, we substitute this into the inequalities derived in Step 1.

- For $\begin{bmatrix} R & B \end{bmatrix}$, we get:

$$\text{rank}\left(\begin{bmatrix} R & B \end{bmatrix}\right) \leq 1 + \text{rank}(B).$$

- For $\begin{bmatrix} R & A \end{bmatrix}$, we get:

$$\text{rank}\left(\begin{bmatrix} R & A \end{bmatrix}\right) \leq 1 + \text{rank}(A).$$

- For $\begin{bmatrix} R & A & C \end{bmatrix}$, we get:

$$\text{rank}\left(\begin{bmatrix} R & A & C \end{bmatrix}\right) \leq 1 + \text{rank}(A) + \text{rank}(C).$$

Step 3: Summing the First Two Terms

Now we add the two inequalities for the ranks of the matrices $\begin{bmatrix} R & B \end{bmatrix}$ and $\begin{bmatrix} R & A \end{bmatrix}$:

$$\text{rank}\left(\begin{bmatrix} R & B \end{bmatrix}\right) + \text{rank}\left(\begin{bmatrix} R & A \end{bmatrix}\right) \leq (1 + \text{rank}(B)) + (1 + \text{rank}(A)) = 2 + \text{rank}(B) + \text{rank}(A).$$

Step 4: Comparing the Terms

Next, we compare the expression we obtained in Step 3 with the inequality for the rank of $\begin{bmatrix} R & A & C \end{bmatrix}$. We have:

$$\text{rank}\left(\begin{bmatrix} R & A & C \end{bmatrix}\right) \leq 1 + \text{rank}(A) + \text{rank}(C).$$

Thus, adding 3 to the right-hand side:

$$\text{rank}\left(\begin{bmatrix} R & A & C \end{bmatrix}\right) + 3 \leq 1 + \text{rank}(A) + \text{rank}(C) + 3 = 4 + \text{rank}(A) + \text{rank}(C).$$

Step 5: Establishing the Desired Inequality

We now compare the sum from Step 3 with the right-hand side from Step 4:

$$2 + \text{rank}(B) + \text{rank}(A) \leq 4 + \text{rank}(A) + \text{rank}(C).$$

Rearranging terms gives:

$$\text{rank}(B) \leq 2 + \text{rank}(C).$$

This completes the proof, as we have shown that

$$\text{rank}\left(\begin{bmatrix} R & B \end{bmatrix}\right) + \text{rank}\left(\begin{bmatrix} R & A \end{bmatrix}\right) \leq \text{rank}\left(\begin{bmatrix} R & A & C \end{bmatrix}\right) + 3.$$

□

C.2.3 Claude

Proof. Let $M = \begin{bmatrix} R & A \\ B & C \end{bmatrix}$ denote the full block matrix. We will use the subadditivity property of matrix rank and properties specific to the case when $\text{rank}(R) \leq 1$.

First, we recall the fundamental inequality for block matrices: for any matrices X and Y of compatible dimensions,

$$\text{rank}(X) + \text{rank}(Y) \leq \text{rank}\begin{bmatrix} X \\ Y \end{bmatrix} + \text{rank}\begin{bmatrix} X & Y \end{bmatrix}.$$

We apply this inequality with $X = R$ and appropriate choices for the remaining blocks. However, we need a more refined approach using the rank inequality for block matrices.

By the rank inequality for partitioned matrices, we have

$$\text{rank}(M) \geq \max\left\{\text{rank}\begin{bmatrix} R \\ B \end{bmatrix}, \text{rank}\begin{bmatrix} R & A \end{bmatrix}\right\}.$$

Now, consider the following key observation. Since $\text{rank}(R) \leq 1$, we can write R in the form $R = uv^T$ where u and v are column vectors (or $R = 0$).

Using the Frobenius rank inequality, for the block matrix M, we have

$$\text{rank}(M) \geq \text{rank}\begin{bmatrix} R \\ B \end{bmatrix} + \text{rank}\begin{bmatrix} R & A \end{bmatrix} - \text{rank}(R).$$

This gives us

$$\operatorname{rank}\begin{bmatrix} R \\ B \end{bmatrix} + \operatorname{rank}\begin{bmatrix} R & A \end{bmatrix} \leq \operatorname{rank}(M) + \operatorname{rank}(R).$$

Since $\operatorname{rank}(R) \leq 1$, we obtain

$$\operatorname{rank}\begin{bmatrix} R \\ B \end{bmatrix} + \operatorname{rank}\begin{bmatrix} R & A \end{bmatrix} \leq \operatorname{rank}(M) + 1.$$

However, we need to establish the bound with $+3$ on the right-hand side. To obtain this stronger bound, we use a more general rank inequality. By applying the rank sum inequality for block matrices and using the constraint $\operatorname{rank}(R) \leq 1$, along with the fact that the rank of a matrix is bounded by its dimensions, we can show that the additional slack of 2 comes from the interaction between the blocks.

More precisely, using the generalized rank inequality and the fact that $\operatorname{rank}(R) \leq 1$ provides additional structure, we have

$$\operatorname{rank}\begin{bmatrix} R \\ B \end{bmatrix} + \operatorname{rank}\begin{bmatrix} R & A \end{bmatrix} \leq \operatorname{rank}(M) + 2\operatorname{rank}(R) + 1 \leq \operatorname{rank}(M) + 3,$$

where the last inequality follows from $\operatorname{rank}(R) \leq 1$. □

C.2.4 Gemini

Proof. Let $M = \begin{bmatrix} R & A \\ B & C \end{bmatrix}$. We first establish the general inequality for block matrices:

$$\operatorname{rank}\left(\begin{bmatrix} R \\ B \end{bmatrix}\right) + \operatorname{rank}\left(\begin{bmatrix} R & A \end{bmatrix}\right) \leq \operatorname{rank}(R) + \operatorname{rank}(M).$$

Let $r = \operatorname{rank}(R)$. There exist invertible matrices P and Q such that $PRQ = \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix}$. Let $\widetilde{A} = PA$ and $\widetilde{B} = BQ$. We partition these matrices conformably with the blocks of the transformed R as $\widetilde{A} = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ and $\widetilde{B} = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$, where A_1 has r rows and B_1 has r columns.

Consider the rank of the row block matrix. Since rank is invariant under multiplication by non-singular matrices:

$$\operatorname{rank}\left(\begin{bmatrix} R & A \end{bmatrix}\right) = \operatorname{rank}\left(P\begin{bmatrix} R & A \end{bmatrix}\begin{bmatrix} Q & 0 \\ 0 & I \end{bmatrix}\right) = \operatorname{rank}\left(\begin{bmatrix} I_r & 0 & A_1 \\ 0 & 0 & A_2 \end{bmatrix}\right).$$

Since the first r rows are linearly independent and the last rows have zeros in the first r columns, the rank is the sum of the rank of the top block and the rank of the bottom block after projecting out the top block. Specifically,

$$\operatorname{rank}\left(\begin{bmatrix} R & A \end{bmatrix}\right) = r + \operatorname{rank}(A_2).$$

Similarly, for the column block matrix:

$$\operatorname{rank}\left(\begin{bmatrix} R \\ B \end{bmatrix}\right) = \operatorname{rank}\left(\begin{bmatrix} P & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} R \\ B \end{bmatrix}Q\right) = \operatorname{rank}\left(\begin{bmatrix} I_r & 0 \\ 0 & 0 \\ B_1 & B_2 \end{bmatrix}\right) = r + \operatorname{rank}(B_2).$$

Now consider the rank of the full matrix M:

$$\operatorname{rank}(M) = \operatorname{rank}\left(\begin{bmatrix} P & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} R & A \\ B & C \end{bmatrix}\begin{bmatrix} Q & 0 \\ 0 & I \end{bmatrix}\right) = \operatorname{rank}\left(\begin{bmatrix} I_r & 0 & A_1 \\ 0 & 0 & A_2 \\ B_1 & B_2 & C \end{bmatrix}\right).$$

We perform elementary row and column operations to eliminate A_1 and B_1 using the pivot I_r. These operations do not change the rank:

$$\operatorname{rank}(M) = \operatorname{rank}\left(\begin{bmatrix} I_r & 0 & 0 \\ 0 & 0 & A_2 \\ 0 & B_2 & C' \end{bmatrix}\right) = r + \operatorname{rank}\left(\begin{bmatrix} 0 & A_2 \\ B_2 & C' \end{bmatrix}\right),$$

where $C' = C - B_1 A_1$. By permuting the columns of the sub-block $\begin{bmatrix} 0 & A_2 \\ B_2 & C' \end{bmatrix}$, we obtain $\begin{bmatrix} A_2 & 0 \\ C' & B_2 \end{bmatrix}$. For any block lower triangular matrix $\begin{bmatrix} X & 0 \\ Y & Z \end{bmatrix}$, the rank satisfies $\operatorname{rank}\begin{bmatrix} X & 0 \\ Y & Z \end{bmatrix} \geq \operatorname{rank}(X) + \operatorname{rank}(Z)$. Therefore:

$$\operatorname{rank}\left(\begin{bmatrix} 0 & A_2 \\ B_2 & C' \end{bmatrix}\right) = \operatorname{rank}\left(\begin{bmatrix} A_2 & 0 \\ C' & B_2 \end{bmatrix}\right) \geq \operatorname{rank}(A_2) + \operatorname{rank}(B_2).$$

Substituting this back into the expression for $\operatorname{rank}(M)$, we get:

$$\operatorname{rank}(M) \geq r + \operatorname{rank}(A_2) + \operatorname{rank}(B_2).$$

We now sum the ranks of the marginal blocks:

$$\operatorname{rank}\left(\begin{bmatrix} R \\ B \end{bmatrix}\right) + \operatorname{rank}\left(\begin{bmatrix} R & A \end{bmatrix}\right) = (r + \operatorname{rank}(B_2)) + (r + \operatorname{rank}(A_2)) = r + (r + \operatorname{rank}(A_2) + \operatorname{rank}(B_2)).$$

Using the inequality derived for $\operatorname{rank}(M)$, we substitute $r + \operatorname{rank}(A_2) + \operatorname{rank}(B_2) \leq \operatorname{rank}(M)$:

$$\operatorname{rank}\left(\begin{bmatrix} R \\ B \end{bmatrix}\right) + \operatorname{rank}\left(\begin{bmatrix} R & A \end{bmatrix}\right) \leq r + \operatorname{rank}(M) = \operatorname{rank}(R) + \operatorname{rank}(M).$$

Given the hypothesis $\operatorname{rank}(R) \leq 1$, we have:

$$\operatorname{rank}\left(\begin{bmatrix} R \\ B \end{bmatrix}\right) + \operatorname{rank}\left(\begin{bmatrix} R & A \end{bmatrix}\right) \leq \operatorname{rank}(M) + 1.$$

Since $1 \leq 3$, it follows that:

$$\operatorname{rank}\left(\begin{bmatrix} R \\ B \end{bmatrix}\right) + \operatorname{rank}\left(\begin{bmatrix} R & A \end{bmatrix}\right) \leq \operatorname{rank}\left(\begin{bmatrix} R & A \\ B & C \end{bmatrix}\right) + 3.$$

□

C.3 Problem 2.3

C.3.1 Problem Statement

Theorem C.3. *The matrices we are working with have $0/1$ entries, so one can view these entries as coming from any field—for instance, we can view them as real numbers, rational numbers, or elements of the finite field $\mathbb{F}_2$. This is important because the value of the rank may depend on the field used.*

(1) Give an example of a 3×3 matrix whose rank over $\mathbb{F}_2$ is not the same as its rank over the reals.

(2) Prove that if M has $0/1$ entries, then the rank of M over the reals is equal to its rank over the rationals, which is at least as large as its rank over $\mathbb{F}_2$.

(3) For every r, show that there is a matrix whose $\mathbb{F}_2$-rank and communication complexity are both r. Conclude that the log-rank conjecture is false over $\mathbb{F}_2$.

C.3.2 GPT

Proof. (1) Consider

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Over $\mathbb{R}$, $\det(A) = 1(0 \cdot 1 - 1 \cdot 1) - 1(1 \cdot 1 - 1 \cdot 0) + 0(1 \cdot 1 - 0 \cdot 0) = -1 - 1 = -2 \neq 0$, hence $\operatorname{rank}_{\mathbb{R}}(A) = 3$. Over $\mathbb{F}_2$, the determinant is computed modulo 2, hence $\det_{\mathbb{F}_2}(A) \equiv -2 \equiv 0 \pmod 2$, so $\operatorname{rank}_{\mathbb{F}_2}(A) \leq 2$. Moreover, the first two rows $(1, 1, 0)$ and $(1, 0, 1)$ are linearly independent over $\mathbb{F}_2$, so $\operatorname{rank}_{\mathbb{F}_2}(A) \geq 2$. Therefore $\operatorname{rank}_{\mathbb{F}_2}(A) = 2 \neq 3 = \operatorname{rank}_{\mathbb{R}}(A)$.

(2) Let M be an $m \times n$ matrix with entries in $\{0, 1\} \subseteq \mathbb{Q} \subseteq \mathbb{R}$. First we show $\operatorname{rank}_{\mathbb{R}}(M) = \operatorname{rank}_{\mathbb{Q}}(M)$. Let $r = \operatorname{rank}_{\mathbb{Q}}(M)$; then there exist r columns of M that are $\mathbb{Q}$-linearly independent, hence also $\mathbb{R}$-linearly independent because $\mathbb{Q} \subseteq \mathbb{R}$, so $\operatorname{rank}_{\mathbb{R}}(M) \geq r$. Conversely, let $s = \operatorname{rank}_{\mathbb{R}}(M)$. By the characterization of rank via minors, there exists an $s \times s$ submatrix B of M with $\det(B) \neq 0$ in $\mathbb{R}$. Since B has integer entries, $\det(B)$ is an integer. An integer that is nonzero in $\mathbb{R}$ is nonzero in $\mathbb{Q}$ as well, hence $\det(B) \neq 0$ in $\mathbb{Q}$, so $\operatorname{rank}_{\mathbb{Q}}(M) \geq s$. Therefore $\operatorname{rank}_{\mathbb{R}}(M) = \operatorname{rank}_{\mathbb{Q}}(M)$.

Now we show $\operatorname{rank}_{\mathbb{Q}}(M) \geq \operatorname{rank}_{\mathbb{F}_2}(M)$. Let $t = \operatorname{rank}_{\mathbb{F}_2}(M)$. Again by the minor characterization over a field, there exists a $t \times t$ submatrix C of M such that $\det(C) \neq 0$ in $\mathbb{F}_2$. Since C has integer entries, its determinant computed over $\mathbb{Z}$ is an integer $d = \det_{\mathbb{Z}}(C)$. The condition $\det_{\mathbb{F}_2}(C) \neq 0$ is equivalent to $d \not\equiv 0 \pmod 2$, hence d is an odd integer, in particular $d \neq 0$ in $\mathbb{Q}$. Therefore $\det(C) \neq 0$ in $\mathbb{Q}$, implying $\operatorname{rank}_{\mathbb{Q}}(M) \geq t$. Combining with the equality already proved yields

$$\operatorname{rank}_{\mathbb{R}}(M) = \operatorname{rank}_{\mathbb{Q}}(M) \geq \operatorname{rank}_{\mathbb{F}_2}(M).$$

(3) Fix $r \in \mathbb{N}$. Let $X = Y = \mathbb{F}_2^r$ and define a Boolean function $f : X \times Y \to \{0, 1\}$ by $f(x, y) = x \cdot y$, where $x \cdot y = \sum_{i=1}^r x_i y_i$ is the standard bilinear form over $\mathbb{F}_2$. Let M_f be the communication matrix of f, indexed by X and Y, with entries in $\mathbb{F}_2$ given by $(M_f)_{x,y} = f(x, y) = x \cdot y$.

We prove $\operatorname{rank}_{\mathbb{F}_2}(M_f) = r$. For each $i \in \{1, \ldots, r\}$ define the column vector $c_i \in \mathbb{F}_2^X$ by $c_i(x) = x_i$. For any $y \in Y$, the y-th column of M_f is the function $x \mapsto x \cdot y = \sum_{i=1}^r y_i x_i = \sum_{i=1}^r y_i c_i(x)$, hence every column lies in the $\mathbb{F}_2$-span of $\{c_1, \ldots, c_r\}$ and therefore $\operatorname{rank}_{\mathbb{F}_2}(M_f) \leq r$. On the other hand, the columns corresponding to the standard basis vectors $e_1, \ldots, e_r \in \mathbb{F}_2^r$ are exactly $c_1, \ldots, c_r$, since $(M_f)_{x,e_i} = x \cdot e_i = x_i = c_i(x)$. These r columns are $\mathbb{F}_2$-linearly independent: if $\sum_{i=1}^r \alpha_i c_i = 0$ as functions on X, then evaluating at $x = e_j$ gives $0 = \sum_{i=1}^r \alpha_i c_i(e_j) = \sum_{i=1}^r \alpha_i (e_j)_i = \alpha_j$ for each j, hence all $\alpha_i = 0$. Thus $\operatorname{rank}_{\mathbb{F}_2}(M_f) \geq r$, and therefore $\operatorname{rank}_{\mathbb{F}_2}(M_f) = r$.

We now show the deterministic communication complexity $D(f)$ equals r. First, $D(f) \leq r$ because Alice can send $x \in \mathbb{F}_2^r$ using r bits and Bob can compute $x \cdot y$. For the lower bound, suppose there is a deterministic protocol computing f with communication cost c. Such a protocol partitions $X \times Y$ into at most 2^c monochromatic rectangles. Fix any monochromatic rectangle $A \times B$ in this partition. If it is monochromatic with value 1, then $0 \notin A$ because $f(0, y) = 0$ for all y, and also $0 \notin B$ because $f(x, 0) = 0$ for all x. If it is monochromatic with value 0, no further restriction is needed. In either case, we claim $|A||B| \leq 2^r \cdot 2^{r-1} = 2^{2r-1}$ for every monochromatic rectangle. Indeed, if $A \times B$ is monochromatic with value 1, pick any $a \in A$. For all $b \in B$ we have $a \cdot b = 1$, so B is contained in the affine hyperplane $\{b \in \mathbb{F}_2^r : a \cdot b = 1\}$, which has size 2^{r-1}. Since $|A| \leq 2^r$, we get $|A||B| \leq 2^r \cdot 2^{r-1}$. If $A \times B$ is monochromatic with value 0, pick any $a \in A$ with $a \neq 0$ if possible; then for all $b \in B$ we have $a \cdot b = 0$, so B is contained in the hyperplane $\{b : a \cdot b = 0\}$ of size 2^{r-1}, giving $|A||B| \leq 2^r \cdot 2^{r-1}$. If $A \subseteq \{0\}$, then $|A||B| \leq 1 \cdot 2^r \leq 2^{2r-1}$ for $r \geq 1$ (and the case $r = 0$ is trivial). Hence every monochromatic rectangle has size at most 2^{2r-1}.

The whole matrix has $|X \times Y| = 2^{2r}$ entries, so at least $2^{2r}/2^{2r-1} = 2$ rectangles are needed; this alone is weak. To obtain the sharp lower bound, apply the same argument symmetrically to both sides via the following refinement. Consider any monochromatic rectangle $A \times B$ with $A \neq \emptyset$ and $B \neq \emptyset$. If there exists $a \in A$ with $a \neq 0$, then B is contained in a hyperplane of size 2^{r-1}, hence $|B| \leq 2^{r-1}$. Similarly, if there exists $b \in B$ with $b \neq 0$, then A is contained in a hyperplane of size 2^{r-1}, hence $|A| \leq 2^{r-1}$. Therefore, unless $A \subseteq \{0\}$ or $B \subseteq \{0\}$, we have $|A||B| \leq 2^{2r-2}$. There are exactly 2^r pairs with $x = 0$

and 2^r pairs with $y = 0$, and their union has size $2^{r+1} - 1 \leq 2^{r+1}$. All remaining pairs, of which there are $2^{2r} - (2^{r+1} - 1)$, must be covered by rectangles satisfying $|A||B| \leq 2^{2r-2}$. Hence the number of such rectangles is at least

$$\frac{2^{2r} - (2^{r+1} - 1)}{2^{2r-2}} = 4 - \frac{2^{r+1} - 1}{2^{2r-2}} \geq 4 - \frac{2^{r+1}}{2^{2r-2}} = 4 - 2^{3-r}.$$

For $r \geq 3$, this is at least 3, still not enough. We therefore use a standard fooling-set argument to obtain a linear lower bound. Let $S = \{(e_i, e_i) : i = 1, \ldots, r\} \subseteq X \times Y$. Then for each i, $f(e_i, e_i) = 1$. For $i \neq j$, we have $f(e_i, e_j) = e_i \cdot e_j = 0$ and $f(e_j, e_i) = 0$. Thus S is a fooling set of size r, meaning that any monochromatic rectangle can contain at most one element of S: if a rectangle contained both (e_i, e_i) and (e_j, e_j) with $i \neq j$, then it would also contain (e_i, e_j) and (e_j, e_i), but $f(e_i, e_i) = f(e_j, e_j) = 1$ while $f(e_i, e_j) = 0$, contradicting monochromaticity. Therefore any deterministic protocol must use at least r monochromatic rectangles, hence $2^c \geq r$ and thus $c \geq \lceil \log_2 r \rceil$. This gives only logarithmic lower bound, so we sharpen by considering an r-bit indexing reduction. Define a restricted input set $X' = \{0, 1\}^r$ and $Y' = \{e_1, \ldots, e_r\}$. On this promise, $f(x, e_i) = x_i$. Any protocol computing f on all inputs in particular computes the index function $g(x, i) = x_i$ by taking Bob's input as e_i. It is standard and can be proved as follows that $D(g) = r$: for each $x \in \{0, 1\}^r$, consider the transcript produced when Alice has input x and Bob has input e_i for each i. If two distinct strings $x \neq x'$ yielded identical transcripts for all i, then Bob's output would be the same on (x, e_i) and (x', e_i) for all i, forcing $x_i = x'_i$ for all i, a contradiction. Hence the set of transcripts determined by Alice's input has size at least 2^r. A protocol with communication cost c has at most 2^c possible transcripts, so $2^c \geq 2^r$ and thus $c \geq r$. Therefore $D(f) \geq D(g) = r$. Together with the upper bound $D(f) \leq r$, we obtain $D(f) = r$.

We have constructed, for each r, a $0/1$ matrix M_f over $\mathbb{F}_2$ with $\mathrm{rank}_{\mathbb{F}_2}(M_f) = r$ and deterministic communication complexity $D(f) = r$. The $\mathbb{F}_2$ version of the log-rank conjecture would assert that $D(f) \leq (\log_2 \mathrm{rank}_{\mathbb{F}_2}(M_f))^{O(1)}$, i.e. bounded by a fixed polynomial in $\log_2 r$, for all such matrices. Our family satisfies $D(f) = r$, which is not bounded by any fixed polynomial in $\log_2 r$ as $r \to \infty$. Hence the log-rank conjecture is false over $\mathbb{F}_2$. □

C.3.3 Claude

Proof. We prove each part of the theorem in turn.

Part (1): We construct a 3×3 matrix with different ranks over $\mathbb{F}_2$ and $\mathbb{R}$.

Consider the matrix

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Over $\mathbb{R}$: We compute the determinant. We have

$$\det(M) = 1(0 \cdot 1 - 1 \cdot 1) - 1(1 \cdot 1 - 1 \cdot 0) + 0(1 \cdot 1 - 0 \cdot 0) = -1 - 1 = -2 \neq 0.$$

Since the determinant is nonzero, M has rank 3 over $\mathbb{R}$.

Over $\mathbb{F}_2$: We perform row reduction. The matrix is

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Adding row 1 to row 2 (in $\mathbb{F}_2$, where $1 + 1 = 0$):

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Adding row 2 to row 3:

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

Thus, the rank over $\mathbb{F}_2$ is 2.

Part (2): Let M be a matrix with $0/1$ entries.

First, we show that $\text{rank}_{\mathbb{Q}}(M) = \text{rank}_{\mathbb{R}}(M)$. The rank of a matrix equals the size of the largest nonzero minor. Since M has entries in $\{0, 1\} \subseteq \mathbb{Q}$, all minors are rational numbers. A minor is nonzero over $\mathbb{Q}$ if and only if it is nonzero over $\mathbb{R}$ (since $\mathbb{Q} \subseteq \mathbb{R}$ and a nonzero rational remains nonzero when viewed as a real number). Therefore, $\text{rank}_{\mathbb{Q}}(M) = \text{rank}_{\mathbb{R}}(M)$.

Next, we show that $\text{rank}_{\mathbb{Q}}(M) \geq \text{rank}_{\mathbb{F}_2}(M)$. Let $r = \text{rank}_{\mathbb{F}_2}(M)$. Then there exist r rows of M that are linearly independent over $\mathbb{F}_2$. Suppose, for contradiction, that these same r rows are linearly dependent over $\mathbb{Q}$. Then there exist rational coefficients $c_1, \ldots, c_r$, not all zero, such that $\sum_{i=1}^{r} c_i v_i = 0$ where v_i are the row vectors.

Clearing denominators, we obtain integer coefficients $a_1, \ldots, a_r$, not all zero, with $\sum_{i=1}^{r} a_i v_i = 0$. Reducing these coefficients modulo 2, we get $\bar{a}_1, \ldots, \bar{a}_r \in \mathbb{F}_2$ with $\sum_{i=1}^{r} \bar{a}_i v_i = 0$ over $\mathbb{F}_2$. Since not all a_i are even (otherwise all a_i would be divisible by arbitrarily high powers of 2, contradicting that they are nonzero after clearing denominators), not all $\bar{a}_i$ are zero. This contradicts the linear independence of the rows over $\mathbb{F}_2$. Therefore, $\text{rank}_{\mathbb{Q}}(M) \geq r = \text{rank}_{\mathbb{F}_2}(M)$.

Part (3): For each $r \geq 1$, consider the $r \times r$ identity matrix I_r over $\mathbb{F}_2$. This matrix has $\mathbb{F}_2$-rank equal to r.

The communication complexity of I_r is exactly r. To see this, note that Alice (holding row index i) and Bob (holding column index j) need to determine whether $i = j$. Alice can send the r-bit string representing i, and Bob can check if it equals j. This requires r bits of communication. Conversely, any protocol must distinguish between all r possible rows, requiring at least $\lceil \log_2 r \rceil$ bits, but more precisely, to compute $I_r(i, j)$ correctly for all inputs, we need at least r bits since there are r distinct rows that must be distinguished.

Since both the $\mathbb{F}_2$-rank and communication complexity equal r, but the log-rank conjecture would predict communication complexity $O((\log r)^c)$ for some constant c, we have $r \notin O((\log r)^c)$ for large r. Therefore, the log-rank conjecture is false over $\mathbb{F}_2$. □

C.3.4 Gemini

Proof. We address the three parts of the theorem sequentially.

(1) Consider the matrix M given by

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

We compute the rank of M over the real numbers $\mathbb{R}$ and over the finite field $\mathbb{F}_2$.

Over $\mathbb{R}$, the determinant of M is

$$\det(M) = 1(0 - 1) - 1(1 - 0) + 0 = -1 - 1 = -2.$$

Since $\det(M) \neq 0$, the matrix is invertible, and thus $\text{rank}_{\mathbb{R}}(M) = 3$.

Over $\mathbb{F}_2$, we perform arithmetic modulo 2. The sum of the three rows is

$$(1, 1, 0) + (1, 0, 1) + (0, 1, 1) = (2, 2, 2) \equiv (0, 0, 0) \pmod 2.$$

Since the rows are linearly dependent, $\text{rank}_{\mathbb{F}_2}(M) < 3$. However, the top-left 2×2 submatrix $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ has determinant $1(0) - 1(1) = -1 \equiv 1 \pmod 2$. Since there is a non-vanishing minor of size 2, $\text{rank}_{\mathbb{F}_2}(M) = 2$.

Thus, $\text{rank}_{\mathbb{R}}(M) \neq \text{rank}_{\mathbb{F}_2}(M)$.

(2) Let M be a matrix with entries in $\{0, 1\}$.

First, we show $\text{rank}_{\mathbb{R}}(M) = \text{rank}_{\mathbb{Q}}(M)$. Since $\mathbb{Q} \subseteq \mathbb{R}$, the columns of M are vectors in $\mathbb{Q}^n \subseteq \mathbb{R}^n$. If a subset of columns is linearly independent over $\mathbb{Q}$, it remains linearly independent over $\mathbb{R}$. Conversely, if the columns are linearly dependent over $\mathbb{R}$, they are linearly dependent over $\mathbb{Q}$ because the linear system

$Mx = 0$ has rational coefficients; thus, if it has a non-trivial real solution, it has a non-trivial rational solution (by Gaussian elimination). Therefore, the dimension of the column space is the same in both fields.

Next, we show $\text{rank}_{\mathbb{Q}}(M) \geq \text{rank}_{\mathbb{F}_2}(M)$. Let $k = \text{rank}_{\mathbb{F}_2}(M)$. By the definition of rank, there exists a $k \times k$ submatrix S of M such that $\det(S) \neq 0$ in $\mathbb{F}_2$. Since the entries of S are integers (0 or 1), we can evaluate the determinant over the integers, denoted $D = \det(S) \in \mathbb{Z}$. The condition that $\det(S) \neq 0$ in $\mathbb{F}_2$ implies $D \equiv 1 \pmod 2$. Consequently, D is an odd integer, which implies $D \neq 0$ in $\mathbb{Z}$ (and thus in $\mathbb{Q}$ and $\mathbb{R}$). Since M contains a $k \times k$ submatrix with a non-zero determinant over $\mathbb{Q}$, we have $\text{rank}_{\mathbb{Q}}(M) \geq k = \text{rank}_{\mathbb{F}_2}(M)$.

Combining these results, we obtain

$$\text{rank}_{\mathbb{R}}(M) = \text{rank}_{\mathbb{Q}}(M) \geq \text{rank}_{\mathbb{F}_2}(M).$$

(3) For any integer $r \geq 1$, consider the Inner Product function $IP_r : \{0,1\}^r \times \{0,1\}^r \to \{0,1\}$ defined by $IP_r(x, y) = \sum_{i=1}^{r} x_i y_i \pmod 2$. Let M be the communication matrix associated with this function, where rows are indexed by $x \in \{0,1\}^r$ and columns by $y \in \{0,1\}^r$, with entries $M_{xy} = \langle x, y \rangle$ over $\mathbb{F}_2$.

We first determine $\text{rank}_{\mathbb{F}_2}(M)$. The entry M_{xy} can be written as the matrix product $x^T y$ where x and y are column vectors. Thus, M can be decomposed as the sum of r outer products:

$$M = \sum_{i=1}^{r} u_i v_i^T,$$

where u_i is the column vector of length 2^r containing the i-th bit of every input x, and $v_i = u_i$. This explicitly writes M as the sum of r matrices of rank 1, so $\text{rank}_{\mathbb{F}_2}(M) \leq r$. To see that the rank is exactly r, consider the $r \times r$ submatrix formed by rows indexed by the standard basis vectors $e_1, \ldots, e_r$ and columns indexed by $e_1, \ldots, e_r$. This submatrix is the identity matrix I_r, which has rank r. Thus, $\text{rank}_{\mathbb{F}_2}(M) = r$.

Next, we consider the deterministic communication complexity $D(M)$. It is a standard result in communication complexity that computing the inner product of two r-bit strings deterministically requires communicating r bits in the worst case. Thus, $D(M) \geq r$. Since Alice can simply send her entire input x to Bob (requiring r bits), $D(M) \leq r$. Therefore, $D(M) = r$.

The log-rank conjecture states that there exists a constant c such that for all matrices M, $D(M) = O((\log \text{rank}(M))^c)$. If we consider the rank over $\mathbb{F}_2$, we have constructed a matrix where $D(M) = r$ and $\text{rank}_{\mathbb{F}_2}(M) = r$. The conjecture would require $r = O((\log r)^c)$, which is false for sufficiently large r. Thus, the log-rank conjecture is false if the rank is taken over $\mathbb{F}_2$. □

C.4 Problem 2.4

C.4.1 Problem Statement

Theorem C.4. *Show that if $f(x, y)$ has fooling set S such that f evaluates to 1 on every point of S. Then, we shall show that* $\text{rank}(M_f) \geq \sqrt{|S|}$.

1. Show that every row and column of M_f can contain at most one element of S. Thus, after rearranging the rows and columns, we can assume that the elements of S lie on the diagonal.

2. Consider the matrix $A = M_f \oplus M_f$. Prove that $\text{rank}(A) \geq |S|$. *Conclude that* $\text{rank}(M_f) \geq \sqrt{|S|}$.

C.4.2 GPT

Proof. We are given a matrix M_f associated with a function $f(x, y)$, where f has a fooling set S such that $f(x, y) = 1$ for all $(x, y) \in S$. We are tasked with proving that $\text{rank}(M_f) \geq \sqrt{|S|}$.

1. Step 1: Show that every row and column of M_f can contain at most one element of S.

Since f evaluates to 1 on every point of S, for each $(x, y) \in S$, the corresponding entry in M_f is 1. Let us assume, for the sake of contradiction, that there exists a row or column of M_f that contains more than one element from S.

Consider the case where two distinct points (x_1, y_1) and (x_2, y_2) are in the same row or column, so $f(x_1, y_1) = f(x_2, y_2) = 1$. This would imply that the corresponding matrix entries $M_f(x_1, y_1)$ and $M_f(x_2, y_2)$ are both 1. However, since M_f is a matrix representation of a function, the structure of M_f

and the assumption that f evaluates to 1 only on points in S dictates that there can be no two such entries in the same row or column. Therefore, each row and each column of M_f can contain at most one element from S.

Thus, after rearranging the rows and columns of M_f, we can assume without loss of generality that all elements of S lie along the diagonal of M_f.

2. Step 2: Consider the matrix $A = M_f \oplus M_f$.

Let A be the matrix formed by taking the direct sum of M_f with itself, i.e., $A = M_f \oplus M_f$. The size of the matrix A is twice that of M_f, and it is block-diagonal, where each block is M_f. Specifically, A is of the form

$$A = \begin{pmatrix} M_f & 0 \\ 0 & M_f \end{pmatrix}.$$

Since the matrix M_f contains the points of the fooling set S along its diagonal, we conclude that A contains the same points along its diagonal. Therefore, the diagonal entries of A correspond to the points in S.

3. Step 3: Prove that $\mathrm{rank}(A) \geq |S|$.

By the structure of the matrix A, we see that the diagonal blocks of A correspond to two copies of the matrix M_f. Since M_f has the fooling set S along its diagonal, the rank of M_f is at least the number of points in S, i.e., $\mathrm{rank}(M_f) \geq |S|$. Therefore, since A is a direct sum of two such matrices, we have

$$\mathrm{rank}(A) = \mathrm{rank}(M_f) + \mathrm{rank}(M_f) \geq 2 \cdot |S|.$$

4. Step 4: Conclude that $\mathrm{rank}(M_f) \geq \sqrt{|S|}$.

From the previous step, we established that $\mathrm{rank}(A) \geq |S|$. Since $A = M_f \oplus M_f$, we know that $\mathrm{rank}(A)$ is twice the rank of M_f. Therefore, we can conclude that

$$\mathrm{rank}(M_f) \geq \sqrt{|S|}.$$

This completes the proof. □

C.4.3 Claude

Proof. Let $f : X \times Y \to \{0, 1\}$ be a Boolean function with communication matrix M_f, where the rows are indexed by X and columns by Y. Let $S \subseteq X \times Y$ be a fooling set such that $f(x, y) = 1$ for all $(x, y) \in S$.

Recall that S is a fooling set if for any two distinct pairs $(x_1, y_1), (x_2, y_2) \in S$, we have either $f(x_1, y_2) = 0$ or $f(x_2, y_1) = 0$.

Part 1: We show that every row and column of M_f contains at most one element of S.

Suppose for contradiction that a single row x contains two elements of S, say (x, y_1) and (x, y_2) where $y_1 \neq y_2$. Since both are in S, we have $f(x, y_1) = 1$ and $f(x, y_2) = 1$. By the fooling set property applied to these two distinct pairs, we must have either $f(x, y_2) = 0$ or $f(x, y_1) = 0$. This contradicts the fact that both equal 1.

Similarly, suppose a single column y contains two elements of S, say (x_1, y) and (x_2, y) where $x_1 \neq x_2$. Since both are in S, we have $f(x_1, y) = 1$ and $f(x_2, y) = 1$. By the fooling set property, we must have either $f(x_1, y) = 0$ or $f(x_2, y) = 0$, which is again a contradiction.

Therefore, each row and column contains at most one element of S. By permuting rows and columns, we can arrange M_f so that the elements of S lie on the diagonal. Let $|S| = s$.

Part 2: Consider the matrix $A = M_f \oplus M_f$, which denotes the direct sum. This is a block diagonal matrix of the form

$$A = \begin{pmatrix} M_f & 0 \\ 0 & M_f \end{pmatrix}.$$

After rearrangement from Part 1, we may assume $S = \{(x_i, y_i) : i = 1, \ldots, s\}$ where these pairs correspond to diagonal entries of M_f. In the matrix A, these s diagonal entries from the first copy of M_f

and the s corresponding diagonal entries from the second copy of M_f give us $2s$ diagonal positions, all equal to 1.

For each $i \in \{1, \ldots, s\}$, consider the i-th row in the first block and the i-th row in the second block of A. These two rows differ only in their block positions. By the fooling set property, for $i \neq j$, we have $f(x_i, y_j) = 0$ or $f(x_j, y_i) = 0$. This ensures that the rows corresponding to elements of S in the first block are linearly independent from those in the second block.

More precisely, the s rows in the first block of A corresponding to S have the form $(r_i, 0)$ where r_i is the i-th row of M_f (after rearrangement), and the s rows in the second block have the form $(0, r_i)$. These $2s$ rows are linearly independent because the fooling set property ensures that the rows $r_1, \ldots, r_s$ are linearly independent in M_f (each has a unique 1 in position i on the diagonal, and by the fooling property, the off-diagonal entries in these rows cannot create linear dependencies).

However, we need only show that $\mathrm{rank}(A) \geq s$. Since A is a direct sum, $\mathrm{rank}(A) = 2 \cdot \mathrm{rank}(M_f)$. Also, the s rows of M_f corresponding to S are linearly independent (each has a 1 in a unique diagonal position corresponding to S, and the fooling property prevents dependencies). Thus $\mathrm{rank}(M_f) \geq s$.

Wait, let me reconsider. We have $\mathrm{rank}(A) = 2 \cdot \mathrm{rank}(M_f)$ and we need $\mathrm{rank}(A) \geq s = |S|$. This gives $2 \cdot \mathrm{rank}(M_f) \geq s$, so $\mathrm{rank}(M_f) \geq s/2$.

Actually, the correct argument is: the s rows corresponding to S in M_f are linearly independent, so $\mathrm{rank}(M_f) \geq s$. Since M_f has dimensions at least $s \times s$ (after rearrangement), and we need $\mathrm{rank}(M_f) \geq \sqrt{s}$, we observe that if $\mathrm{rank}(M_f) = r$, then M_f has at least r rows and r columns, so $s \leq r^2$, giving $r \geq \sqrt{s}$.

Therefore, $\mathrm{rank}(M_f) \geq \sqrt{|S|}$. □

C.4.4 Gemini

Proof. Let $f : X \times Y \to \{0, 1\}$ be a function and M_f be its associated communication matrix. Let $S = \{(x_1, y_1), \ldots, (x_k, y_k)\}$ be a fooling set of size $k = |S|$ such that $f(x, y) = 1$ for all $(x, y) \in S$. By the definition of a fooling set, for any distinct pairs $(x_i, y_i), (x_j, y_j) \in S$, we have $f(x_i, y_j) = 0$ or $f(x_j, y_i) = 0$.

Step 1: We first show that every row and column of M_f contains at most one element of S. Suppose, for the sake of contradiction, that a row x contains two distinct elements of S, say (x, y_i) and (x, y_j) with $i \neq j$. Since both elements are in S, we have $f(x, y_i) = 1$ and $f(x, y_j) = 1$. However, the fooling set property requires that for these distinct pairs, either $f(x, y_j) = 0$ or $f(x, y_i) = 0$. This contradicts the fact that f evaluates to 1 on all elements of S. Thus, no row can contain more than one element of S. A symmetric argument shows that no column can contain more than one element of S.

Since the rows $\{x_1, \ldots, x_k\}$ are distinct and the columns $\{y_1, \ldots, y_k\}$ are distinct, we can rearrange the rows and columns of M_f such that the elements of S appear on the main diagonal of the top-left $k \times k$ submatrix. Let M_S denote this principal submatrix. By construction, the diagonal entries of M_S are all 1 (since $f(x_i, y_i) = 1$), and for any $i \neq j$, the product of the symmetric off-diagonal entries is 0 (since $f(x_i, y_j) f(x_j, y_i) = 0$).

Step 2: Consider the matrix $A = M_f \otimes M_f$ (interpreting $\oplus$ as the Kronecker product $\otimes$ to utilize rank properties). We aim to prove that $\mathrm{rank}(A) \geq |S|$. We observe the following property of the rank of the Kronecker product:

$$\mathrm{rank}(A) = \mathrm{rank}(M_f \otimes M_f) = (\mathrm{rank}(M_f))^2.$$

To lower bound this rank, consider the Hadamard product (entry-wise product) of M_S and its transpose M_S^T, denoted $M_S \circ M_S^T$. The entries of this matrix are given by:

$$(M_S \circ M_S^T)_{ij} = (M_S)_{ij} \cdot (M_S^T)_{ij} = f(x_i, y_j) \cdot f(x_j, y_i).$$

Using the properties of the fooling set S established in Step 1:

- For diagonal entries ($i = j$): $(M_S \circ M_S^T)_{ii} = f(x_i, y_i) \cdot f(x_i, y_i) = 1 \cdot 1 = 1$.
- For off-diagonal entries ($i \neq j$): $(M_S \circ M_S^T)_{ij} = f(x_i, y_j) \cdot f(x_j, y_i) = 0$.

Thus, $M_S \circ M_S^T = I_k$, the $k \times k$ identity matrix. We apply the known rank inequality for the Hadamard product, $\operatorname{rank}(P \circ Q) \leq \operatorname{rank}(P)\operatorname{rank}(Q)$. Applying this to M_S and M_S^T:

$$\operatorname{rank}(I_k) = \operatorname{rank}(M_S \circ M_S^T) \leq \operatorname{rank}(M_S) \cdot \operatorname{rank}(M_S^T).$$

Since $\operatorname{rank}(I_k) = k = |S|$ and $\operatorname{rank}(M_S) = \operatorname{rank}(M_S^T)$, we have:

$$|S| \leq (\operatorname{rank}(M_S))^2.$$

Since M_S is a submatrix of M_f, $\operatorname{rank}(M_S) \leq \operatorname{rank}(M_f)$. Therefore:

$$|S| \leq (\operatorname{rank}(M_f))^2 = \operatorname{rank}(A).$$

This proves $\operatorname{rank}(A) \geq |S|$. Taking the square root of both sides, we conclude:

$$\sqrt{|S|} \leq \operatorname{rank}(M_f).$$

□

C.5 Problem 2.5

C.5.1 Problem Statement

Lemma C.5. *A matrix with $0/1$ entries of rank r has at most 2^r distinct rows, and at most 2^r distinct columns.*

C.5.2 GPT

Proof. Let A be a matrix with entries in $\{0, 1\}$ and $\operatorname{rank}(A) = r$ over $\mathbb{R}$. We first prove the bound on the number of distinct rows.

Let $R \subseteq \mathbb{R}^n$ be the row space of A, so $\dim R = r$. Let $\mathbb{Z}^n$ denote the integer lattice in $\mathbb{R}^n$. Define

$$\phi : R \cap \mathbb{Z}^n \to (\mathbb{Z}/2\mathbb{Z})^n, \qquad \phi(x) = x \bmod 2,$$

where the reduction is taken coordinatewise. The map ϕ is a group homomorphism from the additive group $R \cap \mathbb{Z}^n$ to $(\mathbb{Z}/2\mathbb{Z})^n$.

Choose r linearly independent rows $u_1, \ldots, u_r$ of A. Each u_i lies in $\{0, 1\}^n \subseteq \mathbb{Z}^n$, and the set $\{u_1, \ldots, u_r\}$ is linearly independent over $\mathbb{R}$. Let U be the $r \times n$ matrix whose ith row is u_i. Since the u_i are linearly independent, U has full row rank r. Hence there exists a set of column indices $J = \{j_1, \ldots, j_r\} \subseteq \{1, \ldots, n\}$ such that the $r \times r$ submatrix U_J obtained by restricting to columns in J is invertible over $\mathbb{R}$. In particular, the linear map

$$\pi : R \to \mathbb{R}^r, \qquad \pi(x) = (x_{j_1}, \ldots, x_{j_r})$$

is an isomorphism of real vector spaces: it is surjective because $\pi(u_1), \ldots, \pi(u_r)$ form the rows of U_J and span $\mathbb{R}^r$, and it is injective because $\dim R = r$.

Let $V := R \cap \mathbb{Z}^n$. Then V is a free abelian group of rank r. Indeed, V is a subgroup of $\mathbb{Z}^n$, hence finitely generated and torsion-free, so it is free abelian; moreover, $V \otimes_{\mathbb{Z}} \mathbb{Q} = (R \cap \mathbb{Z}^n) \otimes_{\mathbb{Z}} \mathbb{Q} = R \cap \mathbb{Q}^n$ has $\mathbb{Q}$-dimension r because R is spanned by the integer vectors $u_1, \ldots, u_r$ and these remain linearly independent over $\mathbb{Q}$, whence $\operatorname{rank}_{\mathbb{Z}}(V) = r$.

Consider the induced map

$$\overline{\phi} : V \to (\mathbb{Z}/2\mathbb{Z})^n, \qquad \overline{\phi}(x) = x \bmod 2,$$

which is a group homomorphism. Its kernel is $2V = \{2x : x \in V\}$, because $x \equiv 0 \pmod 2$ coordinatewise if and only if $x = 2y$ for some $y \in \mathbb{Z}^n$, and if moreover $x \in V \subseteq R$ then $y = x/2 \in R$ and $y \in \mathbb{Z}^n$, hence $y \in V$. Thus $\ker(\overline{\phi}) = 2V$.

Therefore $\overline{\phi}$ induces an injective homomorphism of quotients

$$\psi : V/2V \hookrightarrow (\mathbb{Z}/2\mathbb{Z})^n.$$

Since V is free abelian of rank r, we have $V/2V \cong (\mathbb{Z}/2\mathbb{Z})^r$ as abelian groups, and hence $|V/2V| = 2^r$. It follows that $|\psi(V/2V)| = 2^r$, and in particular the image $\overline{\phi}(V) \subseteq (\mathbb{Z}/2\mathbb{Z})^n$ has cardinality at most 2^r.

Now every row of A lies in $\{0,1\}^n \subseteq \mathbb{Z}^n$ and also lies in the row space R, hence belongs to $V = R \cap \mathbb{Z}^n$. For a vector $x \in \{0,1\}^n$, reduction modulo 2 satisfies $\overline{\phi}(x) = x$ when we identify $\{0,1\}^n$ with its image in $(\mathbb{Z}/2\mathbb{Z})^n$. Consequently, the restriction of $\overline{\phi}$ to the set of $0/1$ rows is injective: if $x, y \in \{0,1\}^n$ and $\overline{\phi}(x) = \overline{\phi}(y)$, then $x = y$ coordinatewise. Hence the set of distinct rows of A injects into $\overline{\phi}(V)$, and thus the number of distinct rows of A is at most $|\overline{\phi}(V)| \leq 2^r$.

The bound on the number of distinct columns follows by applying the preceding argument to the transpose matrix A^T, since $\mathrm{rank}(A^\mathsf{T}) = \mathrm{rank}(A) = r$ and the rows of A^T are precisely the columns of A. □

C.5.3 Claude

Proof. Let A be an $m \times n$ matrix with entries in $\{0,1\}$ and rank r. We prove both claims.

Claim 1: A has at most 2^r distinct rows.

Since rank$(A) = r$, the row space of A has dimension r. Therefore, there exist r linearly independent rows that form a basis for the row space. Let these basis rows be $v_1, v_2, \ldots, v_r \in \mathbb{R}^n$.

Every row of A can be expressed as a linear combination of these basis rows. That is, for each row w of A, there exist scalars $c_1, c_2, \ldots, c_r \in \mathbb{R}$ such that

$$w = c_1 v_1 + c_2 v_2 + \cdots + c_r v_r.$$

Since each row of A has entries in $\{0,1\}$, and the basis rows also have entries in $\{0,1\}$, we can work over the field $\mathbb{F}_2 = \{0,1\}$ with arithmetic modulo 2. Over $\mathbb{F}_2$, each coefficient c_i must be in $\{0,1\}$.

Therefore, each row of A is uniquely determined by a choice of coefficients $(c_1, c_2, \ldots, c_r)$ where each $c_i \in \{0,1\}$. The number of such choices is 2^r.

Hence, A has at most 2^r distinct rows.

Claim 2: A has at most 2^r distinct columns.

Since rank$(A) = r$, we have rank$(A^T) = r$, where A^T denotes the transpose of A. The matrix A^T is an $n \times m$ matrix with entries in $\{0,1\}$ and rank r.

The rows of A^T are precisely the columns of A. Applying Claim 1 to the matrix A^T, we conclude that A^T has at most 2^r distinct rows.

Therefore, A has at most 2^r distinct columns.

This completes the proof. □

C.5.4 Gemini

Proof. Let A be an $m \times n$ matrix with entries $A_{ij} \in \{0,1\}$ and let rank$(A) = r$. We first prove the upper bound for the number of distinct rows. Let $R_1, \ldots, R_m \in \mathbb{R}^n$ denote the rows of A, and let $\mathcal{V} = \mathrm{span}(R_1, \ldots, R_m)$ be the row space of A. By definition, $\dim(\mathcal{V}) = r$.

Since the column rank of A equals the row rank, there exists a set of indices $J = \{j_1, \ldots, j_r\} \subseteq \{1, \ldots, n\}$ such that the columns of A indexed by J are linearly independent. Let $\pi_J : \mathcal{V} \to \mathbb{R}^r$ be the projection map defined by restricting a vector to the coordinates in J. That is, for any vector $v = (v_1, \ldots, v_n) \in \mathcal{V}$, $\pi_J(v) = (v_{j_1}, \ldots, v_{j_r})$.

Consider the submatrix A' consisting of the columns of A indexed by J. Since these columns are linearly independent, rank$(A') = r$. The rows of A' are precisely the vectors $\pi_J(R_1), \ldots, \pi_J(R_m)$. The row space of A', denoted $\mathcal{V}'$, is spanned by these projected rows. Since row rank equals column rank, $\dim(\mathcal{V}') = \mathrm{rank}(A') = r$. Note that $\mathcal{V}' = \mathrm{Im}(\pi_J)$. Applying the Rank-Nullity Theorem to the linear map π_J restricted to the domain $\mathcal{V}$, we have

$$\dim(\ker(\pi_J)) = \dim(\mathcal{V}) - \dim(\mathrm{Im}(\pi_J)) = r - r = 0.$$

Thus, the map π_J is an isomorphism from $\mathcal{V}$ to $\mathcal{V}'$. In particular, π_J is injective.

This injectivity implies that for any two rows R_i and R_k, if $R_i \neq R_k$, then $\pi_J(R_i) \neq \pi_J(R_k)$. Therefore, the number of distinct rows in A is equal to the number of distinct projected vectors in the set $\{\pi_J(R_1), \ldots, \pi_J(R_m)\}$. Since the entries of A are in $\{0, 1\}$, every component of a projected vector $\pi_J(R_i)$ must be either 0 or 1. Hence, $\pi_J(R_i) \in \{0, 1\}^r$ for all $i = 1, \ldots, m$.

The cardinality of the set $\{0, 1\}^r$ is 2^r. Consequently, there are at most 2^r distinct vectors in the set $\{\pi_J(R_1), \ldots, \pi_J(R_m)\}$. It follows that there are at most 2^r distinct rows in A.

To prove the statement for columns, consider the transpose matrix A^T. The entries of A^T are in $\{0, 1\}$, and $\text{rank}(A^T) = \text{rank}(A) = r$. The rows of A^T are the columns of A. Applying the result proven above to A^T, we conclude that A^T has at most 2^r distinct rows, which implies that A has at most 2^r distinct columns. □

C.6 Problem 2.6

C.6.1 Problem Statement

Theorem C.6. *For any symmetric matrix $M \in \{0, 1\}^{n \times n}$ with ones in all diagonal entries,*

$$2^c \geq \frac{n^2}{|M|},$$

where c is the deterministic communication complexity of M, and $|M|$ is the number of ones in M.

C.6.2 GPT

Proof. Let $M \in \{0, 1\}^{n \times n}$ be symmetric with $M_{ii} = 1$ for all $i \in [n]$, and let $|M|$ denote the number of ones in M. Consider the deterministic communication problem in which Alice receives $i \in [n]$ and Bob receives $j \in [n]$, and they must output M_{ij}. Let c be the deterministic communication complexity of this problem, so there exists a deterministic protocol Π that communicates at most c bits on every input pair (i, j) and outputs M_{ij}.

For each transcript t of Π, let $R_t \subseteq [n] \times [n]$ be the set of input pairs (i, j) that yield transcript t under Π. It is standard that each R_t is a combinatorial rectangle, i.e., there exist sets $A_t, B_t \subseteq [n]$ such that $R_t = A_t \times B_t$, because in a deterministic protocol the sequence of messages is determined by Alice's input and Bob's input in a way that factors by rounds, and the set of inputs consistent with a fixed transcript is the Cartesian product of the sets of inputs consistent with Alice's and Bob's local views of that transcript.

The protocol partitions $[n] \times [n]$ into at most 2^c rectangles $\{R_t\}_{t \in T}$, where T is the set of all possible transcripts, and $|T| \leq 2^c$. Moreover, since Π is correct, on each rectangle R_t the output of Π is constant, equal either to 0 or to 1. Let $T_1 \subseteq T$ be the set of transcripts on which Π outputs 1, and define the corresponding 1-rectangles R_t for $t \in T_1$. Then the set of 1-inputs of M is exactly the disjoint union of these 1-rectangles:

$$\{(i, j) \in [n] \times [n] : M_{ij} = 1\} = \bigsqcup_{t \in T_1} R_t.$$

In particular,

$$|M| = \sum_{t \in T_1} |R_t|.$$

Since M is symmetric and has $M_{ii} = 1$ for all i, all diagonal pairs (i, i) are 1-inputs. Because the rectangles $\{R_t\}_{t \in T}$ form a partition of $[n] \times [n]$, each diagonal pair (i, i) belongs to exactly one rectangle, and since $M_{ii} = 1$, it must belong to a 1-rectangle. Thus, for every $i \in [n]$ there exists a unique $t(i) \in T_1$ such that $(i, i) \in R_{t(i)}$. For each $t \in T_1$, define

$$D_t := \{i \in [n] : (i, i) \in R_t\}.$$

Then the sets $\{D_t\}_{t \in T_1}$ form a partition of $[n]$, hence

$$\sum_{t \in T_1} |D_t| = n.$$

Fix $t \in T_1$ and write $R_t = A_t \times B_t$. If $i \in D_t$, then $(i,i) \in A_t \times B_t$, so $i \in A_t \cap B_t$. Therefore $D_t \subseteq A_t \cap B_t$, and in particular $|A_t| \geq |D_t|$ and $|B_t| \geq |D_t|$. Hence

$$|R_t| = |A_t||B_t| \geq |D_t|^2.$$

Summing over all $t \in T_1$ yields

$$|M| = \sum_{t \in T_1} |R_t| \geq \sum_{t \in T_1} |D_t|^2.$$

By the Cauchy–Schwarz inequality, for any finite family of real numbers $(x_t)_{t \in T_1}$ one has $\sum_{t \in T_1} x_t^2 \geq \frac{(\sum_{t \in T_1} x_t)^2}{|T_1|}$. Applying this with $x_t = |D_t|$ gives

$$\sum_{t \in T_1} |D_t|^2 \geq \frac{\left(\sum_{t \in T_1} |D_t|\right)^2}{|T_1|} = \frac{n^2}{|T_1|}.$$

Combining the last two displayed inequalities, we obtain

$$|M| \geq \frac{n^2}{|T_1|},$$

so $|T_1| \geq \frac{n^2}{|M|}$. Finally, since $|T_1| \leq |T| \leq 2^c$, it follows that

$$2^c \geq \frac{n^2}{|M|},$$

as required. □

C.6.3 Claude

Proof. Let $M \in \{0,1\}^{n \times n}$ be a symmetric matrix with ones in all diagonal entries, and let c denote the deterministic communication complexity of M. We will establish that $2^c \geq \frac{n^2}{|M|}$, where $|M|$ denotes the number of ones in M.

By the definition of deterministic communication complexity, there exists a deterministic protocol that computes $M(x,y)$ for any inputs $x, y \in \{1, 2, \ldots, n\}$ using at most c bits of communication. This protocol induces a partition of the input space $[n] \times [n]$ into at most 2^c monochromatic rectangles, where each rectangle $R = A \times B$ (with $A, B \subseteq [n]$) satisfies the property that M is constant on R.

Let $\mathcal{R} = \{R_1, R_2, \ldots, R_k\}$ be the collection of monochromatic rectangles corresponding to this protocol, where $k \leq 2^c$. We partition $\mathcal{R}$ into two subcollections: $\mathcal{R}_1$ consisting of rectangles on which M takes the value 1, and $\mathcal{R}_0$ consisting of rectangles on which M takes the value 0.

For each rectangle $R_i = A_i \times B_i \in \mathcal{R}_1$, all entries $M(x,y)$ with $x \in A_i$ and $y \in B_i$ equal 1. Therefore, the number of ones in M satisfies

$$|M| = \sum_{R_i \in \mathcal{R}_1} |A_i| \cdot |B_i|.$$

Since M is symmetric with ones on the diagonal, we have $M(x,x) = 1$ for all $x \in [n]$. This implies that each diagonal entry (x,x) must lie in some rectangle from $\mathcal{R}_1$.

For any rectangle $R_i = A_i \times B_i \in \mathcal{R}_1$, if $(x,x) \in R_i$, then $x \in A_i \cap B_i$. By symmetry of M, if $M(x,y) = 1$, then $M(y,x) = 1$. This structural property, combined with the monochromatic rectangle covering, implies that each rectangle in $\mathcal{R}_1$ contributes to covering the ones in M.

By the Cauchy-Schwarz inequality applied to the areas of rectangles, we have

$$\left(\sum_{R_i \in \mathcal{R}_1} |A_i| \cdot |B_i|\right) \cdot |\mathcal{R}_1| \geq \left(\sum_{R_i \in \mathcal{R}_1} \sqrt{|A_i| \cdot |B_i|}\right)^2.$$

Since the rectangles in $\mathcal{R}$ partition $[n] \times [n]$ and each diagonal element must be covered by some rectangle in $\mathcal{R}_1$, we have that the union of all rectangles in $\mathcal{R}_1$ contains all n diagonal entries. For each $x \in [n]$, there exists at least one rectangle $R_i = A_i \times B_i \in \mathcal{R}_1$ such that $x \in A_i \cap B_i$.

By a counting argument, since each of the n diagonal positions must be covered and there are at most $k \leq 2^c$ rectangles in total, we obtain

$$\sum_{R_i \in \mathcal{R}_1} |A_i \cap B_i| \geq n.$$

For any rectangle $R_i = A_i \times B_i$, by the AM-GM inequality, $|A_i \cap B_i| \leq \sqrt{|A_i| \cdot |B_i|}$. However, to establish the desired bound, we use the fact that the total area covered by ones is $|M|$, and this must be distributed among at most 2^c rectangles.

Since $|\mathcal{R}_1| \leq k \leq 2^c$ and $|M| = \sum_{R_i \in \mathcal{R}_1} |A_i| \cdot |B_i|$, and noting that the diagonal constraint forces sufficient coverage, we have

$$|M| \cdot 2^c \geq |M| \cdot |\mathcal{R}_1| \geq n^2,$$

where the final inequality follows from the fact that the monochromatic rectangles must efficiently cover the n diagonal ones, and by the structure of communication complexity lower bounds for symmetric matrices.

Therefore, $2^c \geq \frac{n^2}{|M|}$. □

C.6.4 Gemini

Proof. Let M be a symmetric matrix in $\{0,1\}^{n \times n}$ with rows and columns indexed by the set $V = \{1, \ldots, n\}$. Let c denote the deterministic communication complexity of M. A deterministic communication protocol for M with cost c corresponds to a partition of the domain $V \times V$ into a set of disjoint combinatorial rectangles $\mathcal{R} = \{R_1, \ldots, R_L\}$, where $L \leq 2^c$.

Since the protocol computes the function defined by M, each rectangle $R_k \in \mathcal{R}$ is monochromatic, meaning that all entries of M restricted to R_k are either all 0 or all 1. Let $\mathcal{K}_1 \subseteq \{1, \ldots, L\}$ be the set of indices corresponding to the 1-rectangles. That is, for every $k \in \mathcal{K}_1$, $M_{ij} = 1$ for all $(i,j) \in R_k$. Conversely, if $M_{ij} = 1$, then (i,j) must belong to exactly one rectangle R_k with $k \in \mathcal{K}_1$.

The total number of ones in M, denoted by $|M|$, is the sum of the sizes of these disjoint 1-rectangles:

$$|M| = \sum_{k \in \mathcal{K}_1} |R_k|.$$

Each rectangle R_k is a product set $A_k \times B_k$, where $A_k, B_k \subseteq V$. Thus, $|R_k| = |A_k| \cdot |B_k|$.

By the hypothesis, M has ones in all diagonal entries. Therefore, for every $i \in V$, the entry (i,i) is equal to 1. This implies that for each i, the pair (i,i) must be contained in some rectangle R_k with $k \in \mathcal{K}_1$. For $(i,i) \in A_k \times B_k$, it must be that $i \in A_k \cap B_k$. Consequently, the union of the intersections of the sides of the 1-rectangles covers the diagonal indices:

$$V = \bigcup_{k \in \mathcal{K}_1} (A_k \cap B_k).$$

Using the subadditivity of the cardinality measure, we have:

$$n = |V| = \left| \bigcup_{k \in \mathcal{K}_1} (A_k \cap B_k) \right| \leq \sum_{k \in \mathcal{K}_1} |A_k \cap B_k|.$$

For any two sets A_k and B_k, the size of their intersection is bounded by the geometric mean of their sizes:

$$|A_k \cap B_k| \leq \min(|A_k|, |B_k|) \leq \sqrt{|A_k| \cdot |B_k|} = \sqrt{|R_k|}.$$

Substituting this into the summation yields:

$$n \leq \sum_{k \in \mathcal{K}_1} \sqrt{|R_k|}.$$

We now apply the Cauchy-Schwarz inequality to the sum on the right-hand side. Let $|\mathcal{K}_1|$ denote the number of 1-rectangles. We view the sum as the dot product of the vector of all ones and the vector with components $\sqrt{|R_k|}$:

$$\left(\sum_{k\in\mathcal{K}_1} 1\cdot\sqrt{|R_k|}\right)^2 \le \left(\sum_{k\in\mathcal{K}_1} 1^2\right)\left(\sum_{k\in\mathcal{K}_1} (\sqrt{|R_k|})^2\right).$$

This simplifies to:

$$\left(\sum_{k\in\mathcal{K}_1} \sqrt{|R_k|}\right)^2 \le |\mathcal{K}_1|\cdot\sum_{k\in\mathcal{K}_1} |R_k|.$$

Using the inequality $n \le \sum_{k\in\mathcal{K}_1}\sqrt{|R_k|}$, we square both sides to obtain $n^2 \le (\sum_{k\in\mathcal{K}_1}\sqrt{|R_k|})^2$. Combining this with the Cauchy-Schwarz result:

$$n^2 \le |\mathcal{K}_1|\cdot\sum_{k\in\mathcal{K}_1}|R_k|.$$

We know that $\sum_{k\in\mathcal{K}_1}|R_k| = |M|$. Furthermore, the number of 1-rectangles $|\mathcal{K}_1|$ is bounded by the total number of leaves in the protocol tree, so $|\mathcal{K}_1| \le L \le 2^c$. Substituting these values gives:

$$n^2 \le 2^c\cdot|M|.$$

Dividing by $|M|$ (which is non-zero since $n \ge 1$ and the diagonal contains ones), we obtain the desired inequality:

$$2^c \ge \frac{n^2}{|M|}.$$

□

C.7 Problem 2.7

C.7.1 Problem Statement

Theorem C.7. *If the rank of a matrix is $r > 1$, then its communication complexity is at most $O(\sqrt{r}\log^2 r)$.*

Lemma C.8. *Any $m\times n$ matrix that has $0/1$ entries and rank $r \ge 0$ must contain a monochromatic submatrix of size at least $mn\cdot 2^{-O(\sqrt{r}\log r)}$.*

C.7.2 GPT

Proof. Let $M \in \{0,1\}^{X\times Y}$ be the communication matrix of a (total) Boolean function $f : X\times Y \to \{0,1\}$, and let $\mathrm{rank}(M) = r > 1$ over $\mathbb{R}$. We prove that the deterministic communication complexity $D(f)$ is $O(\sqrt{r}\log^2 r)$. Throughout, all logarithms are base 2.

For any submatrix $A = M[X_0\times Y_0]$ induced by nonempty $X_0 \subseteq X$ and $Y_0 \subseteq Y$, we write $|A| = |X_0||Y_0|$ and $r(A) = \mathrm{rank}(A)$. Clearly $r(A) \le r$.

We use the following consequence of the stated lemma. There exists an absolute constant $c \ge 1$ such that for every $0/1$ matrix A with $r(A) = s \ge 1$ there is a monochromatic rectangle $R = U\times V$ inside A with

$$|U||V| \ge |A|\cdot 2^{-c\sqrt{s}\log s}.$$

Fix such a constant c once and for all.

Define, for integers $s \ge 1$, the quantity

$$\alpha(s) = 2^{-c\sqrt{s}\log s}.$$

We will prove by induction on $s \ge 1$ that any $0/1$ matrix A with $r(A) \le s$ admits a deterministic protocol that computes A with at most

$$C(s) = K\sqrt{s}\,\log^2 s$$

bits of communication, for a sufficiently large absolute constant K. This implies the theorem by taking $A = M$ and $s = r$.

We need two standard facts. First, if a matrix A has rank at most s, then for any partition of its rows $X_0 = X_1 \sqcup X_2$ and columns $Y_0 = Y_1 \sqcup Y_2$ one has

$$r(A) \le r(A[X_1 \times Y_0]) + r(A[X_2 \times Y_0]), \qquad r(A) \le r(A[X_0 \times Y_1]) + r(A[X_0 \times Y_2]),$$

since the row (respectively column) space is spanned by the union of the corresponding subspaces. Second, if $R = U \times V$ is a monochromatic rectangle in A (say all entries equal to $b \in \{0, 1\}$), then the restriction $A[U \times V]$ has rank at most 1, because it equals either the all-0 matrix (rank 0) or the all-1 matrix (rank 1).

We now describe a protocol template for a given submatrix $A = M[X_0 \times Y_0]$ with $r(A) = s \ge 2$. Apply the lemma to A to obtain a monochromatic rectangle $R = U \times V$ of relative size at least $\alpha(s)$ inside A. Let b be its monochromatic value. Consider the following decision procedure: Alice sends one bit indicating whether $x \in U$. If $x \notin U$, the players recurse on the submatrix $A_1 = A[(X_0 \setminus U) \times Y_0]$. If $x \in U$, Bob sends one bit indicating whether $y \in V$. If $y \in V$, they output b and halt. If $y \notin V$, they recurse on the submatrix $A_2 = A[U \times (Y_0 \setminus V)]$. Thus, each such step uses at most 2 bits, and either halts (inside R) or recurses on A_1 or A_2.

To complete the induction we must argue that along any root-to-leaf path, after not too many steps, the rank parameter strictly decreases in a controlled way, and the total number of steps is bounded solely in terms of s.

Let A and $R = U \times V$ be as above, with $A[U \times V]$ monochromatic. Write $X_0 = (X_0 \setminus U) \sqcup U$ and $Y_0 = V \sqcup (Y_0 \setminus V)$. Consider the block decomposition

$$A = \begin{pmatrix} A[(X_0 \setminus U) \times V] & A[(X_0 \setminus U) \times (Y_0 \setminus V)] \\ A[U \times V] & A[U \times (Y_0 \setminus V)] \end{pmatrix}.$$

By subadditivity of rank under row partition,

$$r(A) \le r\big(A[(X_0 \setminus U) \times Y_0]\big) + r\big(A[U \times Y_0]\big).$$

Also, by subadditivity under column partition applied to $A[U \times Y_0]$,

$$r\big(A[U \times Y_0]\big) \le r\big(A[U \times V]\big) + r\big(A[U \times (Y_0 \setminus V)]\big) \le 1 + r(A_2),$$

since $r(A[U \times V]) \le 1$ and $A_2 = A[U \times (Y_0 \setminus V)]$. Combining, we obtain

$$s = r(A) \le r(A_1) + 1 + r(A_2),$$

where $A_1 = A[(X_0 \setminus U) \times Y_0]$. Therefore,

$$\max\{r(A_1), r(A_2)\} \ge \frac{s-1}{2}.$$

In particular, at least one of A_1 and A_2 has rank at most $\lceil (2s+1)/3 \rceil$ unless the other already has rank at most $\lceil (2s+1)/3 \rceil$; more simply, there always exists $i \in \{1, 2\}$ such that

$$r(A_i) \le s - 1 \quad \text{and} \quad r(A_i) \le \left\lceil \frac{2s+1}{2} \right\rceil = s$$

holds trivially, while the inequality $s \le r(A_1) + 1 + r(A_2)$ ensures that it is impossible that both $r(A_1) = s$ and $r(A_2) = s$ unless $s \le 2s + 1$ (which is always true); hence to enforce strict progress we use the size guarantee of R to iterate a bounded number of times and then apply a rank drop argument as follows.

Fix A with rank $s \ge 2$. Consider repeating the above step, each time applying the lemma to the current submatrix, until either the protocol halts inside the found monochromatic rectangle, or the current submatrix has rank at most $\lfloor s/2 \rfloor$. We claim that after at most

$$T(s) = \left\lceil \frac{2c\sqrt{s}\log s}{\log 2} \right\rceil = O(\sqrt{s}\log s)$$

non-halting steps, the rank must drop to at most $\lfloor s/2 \rfloor$ on every remaining branch. Indeed, suppose we follow a branch that never halts inside the monochromatic rectangle at any of the first t steps. At step j we are at a submatrix $A^{(j)}$ with rank $r(A^{(j)}) \leq s$ and find a monochromatic rectangle $R^{(j)} = U^{(j)} \times V^{(j)}$ in $A^{(j)}$ of relative size at least $\alpha(s)$, since $\alpha(\cdot)$ is nonincreasing and $r(A^{(j)}) \leq s$. By construction, if we do not halt in $R^{(j)}$, then we recurse either to $A^{(j+1)} = A^{(j)}[(X^{(j)} \setminus U^{(j)}) \times Y^{(j)}]$ or to $A^{(j+1)} = A^{(j)}[U^{(j)} \times (Y^{(j)} \setminus V^{(j)})]$. In either case, the set of entries removed from $A^{(j)}$ contains the whole rectangle $R^{(j)}$, so

$$|A^{(j+1)}| \leq |A^{(j)}| - |R^{(j)}| \leq (1 - \alpha(s))|A^{(j)}|.$$

Thus after t non-halting steps,

$$|A^{(t)}| \leq (1 - \alpha(s))^t |A^{(0)}|.$$

On the other hand, if $r(A^{(t)}) > s/2$, then by the lemma applied to $A^{(t)}$ there exists a monochromatic rectangle in $A^{(t)}$ of size at least $\alpha(s/2)\,|A^{(t)}| \geq \alpha(s)\,|A^{(t)}|$, hence $|A^{(t)}|$ must be at least 1 and in particular cannot be smaller than $2^{-c\sqrt{s}\log s}|A^{(0)}|$ once t exceeds a constant multiple of $\sqrt{s}\log s$. Concretely, using $(1-\alpha)^t \leq e^{-\alpha t}$, if $t \geq \alpha(s)^{-1} c\sqrt{s}\log s$ then $|A^{(t)}| \leq e^{-c\sqrt{s}\log s}|A^{(0)}| < 2^{-c\sqrt{s}\log s}|A^{(0)}|$, which contradicts the existence of a rectangle of relative size $\alpha(s)$ inside $A^{(t)}$ unless $r(A^{(t)}) \leq s/2$. Therefore choosing $T(s) = \lceil c\sqrt{s}\log s/\alpha(s) \rceil$ suffices; since $\alpha(s) = 2^{-c\sqrt{s}\log s}$, this gives $T(s) = 2^{O(\sqrt{s}\log s)}$, and hence in particular $T(s) = O(\sqrt{s}\log s)$ after absorbing constants into c and redefining $\alpha(s)$ accordingly. Fixing such constants, we may assume $T(s) \leq K_0\sqrt{s}\log s$ for an absolute K_0.

Therefore, we can build a protocol in phases. In each phase with current rank parameter s, we spend at most $2T(s) \leq 2K_0\sqrt{s}\log s$ bits applying the above rectangle test repeatedly; either we halt, or we reach a submatrix of rank at most $\lfloor s/2 \rfloor$ and begin a new phase with rank parameter at most $\lfloor s/2 \rfloor$.

Let $s_0 = r$ and $s_{j+1} = \lfloor s_j/2 \rfloor$ until $s_L \leq 1$, so $L \leq \lceil \log r \rceil$. The total communication on any root-to-leaf path is at most

$$\sum_{j=0}^{L-1} 2K_0\sqrt{s_j}\log s_j + O(1) \leq 2K_0 \sum_{j=0}^{\infty} \sqrt{r/2^j}\,\log r + O(1) \leq 2K_0 \log r \cdot \sqrt{r} \sum_{j=0}^{\infty} 2^{-j/2} + O(1).$$

Since $\sum_{j=0}^{\infty} 2^{-j/2}$ converges to an absolute constant, there exists an absolute constant K_1 such that the total is at most $K_1\sqrt{r}\log r + O(1)$. Enlarging constants if necessary and noting that $\log r \leq \log^2 r$ for all $r \geq 2$, we obtain

$$D(f) \leq K\sqrt{r}\,\log^2 r$$

for some absolute constant K, as required.

This completes the induction and hence proves the theorem. □

C.7.3 Claude

Proof. We prove the theorem by establishing a connection between the rank of a matrix and its communication complexity through the existence of large monochromatic rectangles.

Let M be an $m \times n$ matrix with $0/1$ entries and rank $r > 1$. We proceed in several steps.

Step 1: Application of the Lemma. By the given lemma, the matrix M contains a monochromatic submatrix (rectangle) of size at least $mn \cdot 2^{-O(\sqrt{r}\log r)}$. Let this submatrix have dimensions $m' \times n'$ where $m' \cdot n' \geq mn \cdot 2^{-O(\sqrt{r}\log r)}$.

Step 2: Rectangle Covering Protocol. The communication complexity of M can be bounded by constructing a protocol that covers the matrix with monochromatic rectangles. Each monochromatic rectangle of size $a \times b$ can be communicated using $O(\log m + \log n)$ bits: Alice sends the row indices (using $O(\log m)$ bits) and Bob sends the column indices (using $O(\log n)$ bits), plus one bit for the color.

Step 3: Counting Required Rectangles. Let $s = 2^{-O(\sqrt{r}\log r)}$ be the guaranteed fraction from the lemma. At each stage of the covering process, we can find a monochromatic rectangle covering at least an s fraction of the remaining entries. Therefore, after k rectangles, we have covered at least $1 - (1-s)^k$ of the matrix entries.

To cover all mn entries, we need $k = O(\log(mn)/s) = O(\log(mn) \cdot 2^{O(\sqrt{r}\log r)})$ rectangles.

Step 4: Bounding the Number of Rectangles. Since $2^{O(\sqrt{r}\log r)} = 2^{c\sqrt{r}\log r}$ for some constant $c > 0$, we have:

$$2^{c\sqrt{r}\log r} = r^{c\sqrt{r}} = e^{c\sqrt{r}\ln r} \leq e^{c\sqrt{r}\cdot\sqrt{r}} = e^{cr}$$

for sufficiently large r. However, we use a more refined analysis.

Note that $2^{c\sqrt{r}\log r} = (2^{\log r})^{c\sqrt{r}} = r^{c\sqrt{r}}$. For the communication complexity, we need at most $O(r^{c\sqrt{r}})$ rectangles when m, n are polynomial in r.

Step 5: Improved Bound via Iterative Covering. Using a more sophisticated covering argument, we partition the matrix recursively. At depth d of the recursion, we have submatrices of rank at most $r/2^d$. By the lemma, each submatrix at depth d contains a monochromatic rectangle of relative size $2^{-O(\sqrt{r/2^d}\log(r/2^d))}$.

The recursion terminates when the rank becomes constant, which occurs at depth $d = O(\log r)$. At each level, we communicate $O(\log m + \log n)$ bits per rectangle found.

The total number of rectangles needed is bounded by analyzing the recursion tree. The key observation is that the total communication is:

$$\sum_{d=0}^{O(\log r)} 2^d \cdot O(\sqrt{r/2^d}\log r) \cdot O(\log(mn))$$

This sum is dominated by the terms where d is small, giving:

$$O(\sqrt{r}\log r \cdot \log(mn)) = O(\sqrt{r}\log^2 r)$$

when $m, n \leq 2^{O(r)}$.

Step 6: Conclusion. Therefore, the communication complexity of M is at most $O(\sqrt{r}\log^2 r)$, as required. □

C.7.4 Gemini

Proof. Let M be an $m \times n$ matrix of rank r. We aim to bound the deterministic communication complexity $D(M)$.

First, we invoke the provided Lemma. The Lemma states that any $0/1$ matrix of rank at most r contains a monochromatic submatrix (rectangle) of relative measure at least $2^{-O(\sqrt{r}\log r)}$. Let $\delta(r) = 2^{-C\sqrt{r}\log r}$ for some constant $C > 0$ be this lower bound on the density of the largest monochromatic rectangle.

We utilize a standard reduction from the density of monochromatic rectangles to communication complexity, specifically the protocol strategy analyzed by Nisan and Wigderson (1995). The proof proceeds in two main steps: bounding the size of a monochromatic rectangle cover, and then bounding the communication cost to identify a rectangle in that cover.

Step 1: Constructing a Monochromatic Cover

Let S be the set of entries of the matrix M. We construct a cover of S by monochromatic rectangles using a greedy algorithm. In each step i, let S_i be the set of entries not yet covered (initially $S_0 = S$). The submatrix corresponding to S_i has rank at most r (as it is a sub-pattern of M, though technically the zero-pattern outside S_i might increase rank, we consider the restriction of the function). More formally, we apply the Lemma iteratively. While S_i is not empty, the Lemma guarantees the existence of a monochromatic rectangle $R_{i+1} \subseteq S_i$ such that

$$|R_{i+1}| \geq \delta(r) \cdot |S_i|.$$

We add R_{i+1} to our cover and set $S_{i+1} = S_i \setminus R_{i+1}$. The size of the uncovered portion decreases geometrically:

$$|S_{i+1}| \leq (1 - \delta(r))|S_i|.$$

To cover all mn entries (i.e., reduce the uncovered set to size < 1), the number of rectangles t required satisfies:

$$(1 - \delta(r))^t \cdot mn < 1 \implies t \leq O\left(\frac{1}{\delta(r)}\log(mn)\right).$$

Without loss of generality, for a communication matrix of rank r, we can assume the number of distinct rows and columns is at most 2^r (otherwise, there are identical rows/columns which can be handled without communication). Thus, $\log(mn) \leq O(r)$. The size of the cover $\mathcal{C} = \{R_1, \ldots, R_t\}$ is bounded by:

$$t \leq O\left(\frac{r}{\delta(r)}\right) = O\left(r \cdot 2^{C\sqrt{r}\log r}\right) = 2^{O(\sqrt{r}\log r)}.$$

Step 2: The Protocol

Alice and Bob possess the cover $\mathcal{C} = \{R_1, \ldots, R_t\}$ (which is known to both as it depends only on M). Their goal is to find an index i such that $(x, y) \in R_i$. Since $\mathcal{C}$ covers M, such an i exists. This problem is equivalent to computing a function with a cover size t. A known result by Nisan and Wigderson states that for a matrix of rank r, the communication complexity of finding a covering rectangle is bounded by $O(\log t \cdot \log r)$.

The protocol operates as follows: The players construct a binary search tree over the indices $\{1, \ldots, t\}$. At each node of the tree, they need to determine if the solution lies in the left or right subtree. This effectively requires determining if $(x, y) \in \bigcup_{j \in L} R_j$ for some subset of rectangles L. While the union of rectangles is not a rectangle, the rank constraint allows efficient checking. Specifically, the protocol for finding a leaf in a cover of size t for a rank r matrix has complexity $O(\log t \cdot \log r)$.

Substituting the bound for t:

$$\log t = O(\sqrt{r}\log r).$$

The total communication complexity is therefore:

$$D(M) \leq O(\log t \cdot \log r) \leq O\left((\sqrt{r}\log r) \cdot \log r\right) = O(\sqrt{r}\log^2 r).$$

Thus, the communication complexity of M is at most $O(\sqrt{r}\log^2 r)$. □

D Problems in Chapter 3

D.1 Problem 3.1

D.1.1 Problem Statement

Lemma D.1. *In this exercise, we design a randomized protocol for finding the first difference between two n-bit strings. Alice and Bob are given n bit strings $x \neq y$ and want to find the smallest i such that $x_i \neq y_i$. We already saw how to accomplish this using $O(\log n \log\log n)$ bits of communication. Here we do it with $O(\log n)$ bits of communication. For simplicity, assume that n is a power of two.*

Define a rooted binary tree as follows. Every vertex v in the tree corresponds to an interval I_v of coordinates from $[n]$. The root corresponds to the interval $I_{root} = [n]$. Every leaf in the tree corresponds to an interval of size 1. Every internal vertex v has two children. The left child corresponds to the left half of I_v, and the right child corresponds to the right half of I_v.

Extend the tree as follows. At each leaf of the binary tree, attach a path of length $3\log n$. Every vertex of this path represents the same interval of size 1. The depth of the overall tree is now $4\log n$.

Fill in the details of the following protocol. The parties use their inputs and hashing to start at the root of the tree and try to navigate to the smallest interval that contains the index i that they seek. In each step, the parties either move to a parent or move to a child of the node that they are currently at. When the parties are at a vertex v that corresponds to the interval I_v, they exchange $O(1)$ hash bits to confirm that the first difference lies in I_v. If this hash shows that the first difference does not lie in I_v, they move to the parent of the current node. Otherwise, they exchange $O(1)$ hash bits to decide to which child of v to move to.

Prove an upper bound on the expected number of bits communicated and a lower bound on the success probability:

1. Argue that as long as the number of nodes where the protocol made the correct choice exceeds the number of nodes where the parties made the wrong choice by $\log n$, the protocol you defined succeeds in computing i.

2. Use the Chernoff-Hoeffding bound to argue that the number of hashes that give the correct answer is high enough to ensure that the protocol succeeds with high probability on any input.

D.1.2 GPT

Proof. Let $x, y \in \{0,1\}^n$ with $x \neq y$, and let $i^* = \min\{i \in [n] : x_i \neq y_i\}$. For an interval $I \subseteq [n]$, define the predicate

$$D(I) = \mathbf{1}\big[\exists j \in I \text{ such that } x_j \neq y_j\big],$$

and define similarly the predicate

$$F(I) = \mathbf{1}\big[\exists j \in I \text{ such that } j \leq i^* \text{ and } x_j \neq y_j\big].$$

By definition of i^*, for every interval I we have $F(I) = \mathbf{1}[i^* \in I]$, and $F(I) = 1$ implies $D(I) = 1$.

For each node v of the extended tree, let I_v be its associated interval. The extended tree has depth $4 \log n$: the original balanced binary tree has depth $\log n$ and each leaf is extended by a downward path of length $3 \log n$ whose nodes all have the same singleton interval.

Fix an error parameter $p \in (0, 1/4)$, and fix a family of public-coin hash functions as follows. For each query of an interval $I \subseteq [n]$, the parties sample independent public randomness r and compute a single-bit hash

$$h_r(x|_I) = \langle r, x|_I \rangle \bmod 2, \qquad h_r(y|_I) = \langle r, y|_I \rangle \bmod 2,$$

where r is a uniformly random binary vector of the same length as $x|_I$ and $\langle \cdot, \cdot \rangle$ is the mod-2 inner product. Then $h_r(x|_I) \oplus h_r(y|_I) = \langle r, x|_I \oplus y|_I \rangle$ equals 0 with probability 1 if $x|_I = y|_I$, and equals 1 with probability $1/2$ otherwise. Repeating independently t times and taking the OR of the t parity differences yields a one-bit test $\mathsf{NE}_t(I)$ with the properties

$$\Pr[\mathsf{NE}_t(I) = 1 \mid x|_I \neq y|_I] = 1 - 2^{-t}, \qquad \Pr[\mathsf{NE}_t(I) = 1 \mid x|_I = y|_I] = 0.$$

Set t so that $2^{-t} \leq p$, e.g. $t = \lceil \log_2(1/p) \rceil = O(1)$. Each execution of $\mathsf{NE}_t(I)$ requires t bits from Alice to Bob and t bits from Bob to Alice (or equivalently exchange of the two hash values), hence $O(1)$ communication.

Define the protocol as a walk on the extended tree. Maintain a current node v. Initially v is the root. At each step, do the following. First perform the confirmation test at v: compute $\mathsf{NE}_t(I_v)$. If $\mathsf{NE}_t(I_v) = 0$, move to the parent of v (if v is the root, stay at the root). If $\mathsf{NE}_t(I_v) = 1$, then if v is on the attached path below a leaf (so that I_v is a singleton), move to the unique child (down the path). Otherwise v is an internal node of the original binary tree with left and right children v_L, v_R, and the parties must decide which child contains i^*. They perform the decision test on the left child interval: compute $\mathsf{NE}_t(I_{v_L})$. If $\mathsf{NE}_t(I_{v_L}) = 1$, move to v_L; otherwise move to v_R. The protocol stops when it reaches depth $4 \log n$ (the end of the attached path) and outputs the unique index in the singleton interval there.

For each node v, call v good if $i^* \in I_v$, and bad otherwise. Note that for every node v on the unique root-to-leaf path in the original binary tree to the leaf representing $\{i^*\}$, v is good, and all siblings of nodes on this path are bad; additionally every node on the attached path below the leaf $\{i^*\}$ is good and every node on attached paths below other leaves is bad.

Define a step of the walk to be correct if the move is consistent with the ideal deterministic navigation toward the leaf $\{i^*\}$ and then down its attached path; otherwise the step is wrong. More explicitly: if the current node v is good, then the correct action is not to move to the parent, and if v is an internal node of the original tree then the correct child is the unique child whose interval contains i^*; if v is on the attached path below $\{i^*\}$, the correct action is to move to the child along the path. If the current node v is bad, then the correct action is to move to the parent (since $i^* \notin I_v$).

Each step uses at most two tests $\mathsf{NE}_t(\cdot)$, hence communicates $O(1)$ bits. It remains to bound the expected number of steps and the success probability.

Define the potential function $\Phi(v)$ to be the graph distance in the extended tree from v to the unique terminal node z at depth $4 \log n$ on the attached path below the leaf $\{i^*\}$. Let $d = \Phi(v)$. Observe that $0 \leq d \leq 4 \log n$ for all nodes v.

For any node v, consider the effect of one step on Φ. If v is good, then the correct move (to the correct child, or down the attached path) decreases Φ by exactly 1, while the wrong move either goes to the parent or to the wrong child, both of which increase Φ by exactly 1. If v is bad, then the correct move (to the

parent) decreases Φ by exactly 1, while the wrong move (to a child) increases Φ by exactly 1. Therefore, in every step, a correct move decreases Φ by 1 and a wrong move increases Φ by 1.

Let C be the number of correct steps taken and W the number of wrong steps taken in the entire execution up to termination at depth $4 \log n$ (or equivalently until reaching z). Starting from the root, $\Phi(\text{root}) \leq 4 \log n$ and $\Phi(z) = 0$. Since each correct step changes Φ by -1 and each wrong step changes it by $+1$, we have

$$0 = \Phi(z) = \Phi(\text{root}) - C + W,$$

whenever the protocol reaches z and stops there. Rearranging yields $C - W = \Phi(\text{root}) \geq 0$. Conversely, if at some time the walk is at a node v with $\Phi(v) = 0$ then it must be at z and outputs i^*.

For the first requirement, suppose that at some time during the protocol, the inequality $C - W \geq \log n$ holds, where C, W count steps taken so far. Then, since $\Phi(\text{root}) \leq 4 \log n$, we have

$$\Phi(v) = \Phi(\text{root}) - C + W \leq 4 \log n - (C - W) \leq 3 \log n.$$

Now note that any node u with $\Phi(u) \leq 3 \log n$ must lie on the attached path below the leaf $\{i^*\}$, because every node not on that attached path is at graph distance at least $3 \log n + 1$ from z: indeed, any node outside that attached path must first reach the corresponding leaf of the original tree (distance at most $\log n$ from the root) and then traverse at least $3 \log n$ edges to reach the end of some attached path, and to get from the end of any other attached path to z one must go up at least $3 \log n$ edges back to the leaf and then follow at least one edge in the original tree before reaching the correct leaf, hence the distance to z is at least $3 \log n + 1$. Consequently, once $\Phi(v) \leq 3 \log n$, the current node v must be on the correct attached path and hence has singleton interval $\{i^*\}$; from that point onward, regardless of subsequent steps, the interval remains $\{i^*\}$ because every node on the attached path has the same singleton interval. Therefore the output index is necessarily i^*. This proves that if the number of correct steps exceeds the number of wrong steps by at least $\log n$ at termination, then the protocol outputs i^*.

We now prove the second requirement. Consider a single step when the protocol is at a node v that is good and is an internal node in the original binary tree. The step comprises a confirmation test on I_v and a decision test on I_{v_L}. The confirmation test can be wrong only by producing 0 when $D(I_v) = 1$, which happens with probability at most p. Conditioned on confirmation being correct, the decision test can be wrong only if it yields 0 when $D(I_{v_L}) = 1$ or yields 1 when $D(I_{v_L}) = 0$. The latter never happens since NE_t has no false positives, and the former happens with probability at most p. Hence in this case the probability that the step is wrong is at most $2p$.

If v is good and lies on the attached path below $\{i^*\}$, then only a confirmation test on the singleton interval $\{i^*\}$ is performed, and it is wrong with probability at most p.

If v is bad, then the step uses only the confirmation test on I_v. Since v is bad, $F(I_v) = 0$, but it may still be that $D(I_v) = 1$ (if I_v contains some later disagreement). In either case, the correct action is to move to the parent, which occurs if and only if the confirmation test outputs 0. The test outputs 0 with probability 1 when $D(I_v) = 0$, and with probability at least $1 - p$ when $D(I_v) = 1$. Therefore, in all cases the probability the step is wrong is at most p.

Thus, in every step, conditional on the past, the probability that the step is correct is at least $1 - 2p$. Fix $p = 1/16$, so that $1 - 2p = 7/8$.

Let T be the total number of steps executed until termination at depth $4 \log n$. We first show that T is $O(\log n)$ in expectation. Consider the process Φ_t of the potential after t steps. For each step, we have $\Phi_{t+1} - \Phi_t = -1$ with probability at least $7/8$ and $\Phi_{t+1} - \Phi_t = +1$ otherwise, except that Φ_t is bounded below by 0 and the process stops at 0. Therefore, for t before stopping,

$$\mathbb{E}[\Phi_{t+1} - \Phi_t \mid \mathcal{F}_t] \leq -(7/8) + (1/8) = -3/4,$$

where $\mathcal{F}_t$ is the sigma-field of the history up to time t. By optional stopping applied to the supermartingale $\Phi_t + (3/4)t$ stopped at the hitting time τ of 0, and using $\Phi_0 \leq 4 \log n$ and $\Phi_\tau = 0$, we obtain

$$0 + \frac{3}{4}\mathbb{E}[\tau] \leq \mathbb{E}[\Phi_\tau] + \frac{3}{4}\mathbb{E}[\tau] = \mathbb{E}[\Phi_0] \leq 4 \log n,$$

hence $\mathbb{E}[T] = \mathbb{E}[\tau] \leq \frac{16}{3} \log n = O(\log n)$. Since each step communicates $O(1)$ bits, the expected communication is $O(\log n)$.

For the success probability, note that on the event that the protocol terminates by reaching z, we have $C - W = \Phi(\text{root}) \leq 4 \log n$. However, the protocol may terminate at depth $4 \log n$ even if it has not reached z; by the first part, it suffices to ensure that at the time of termination, $C - W \geq \log n$.

Let X_t be the indicator that step t is correct, so $X_t \in \{0, 1\}$ and $\Pr[X_t = 1 \mid \mathcal{F}_{t-1}] \geq 7/8$. Define $S_T = \sum_{t=1}^{T} X_t$ and note that $C = S_T$ and $W = T - S_T$, hence

$$C - W = 2S_T - T.$$

Therefore, the success condition $C - W \geq \log n$ is implied by $S_T \geq (T + \log n)/2$.

We upper bound the failure probability by conditioning on a deterministic horizon. Let $m = 32 \log n$. By Markov's inequality applied to the nonnegative random variable T and the expectation bound $\mathbb{E}[T] \leq (16/3) \log n$, we have

$$\Pr[T > m] \leq \frac{\mathbb{E}[T]}{m} \leq \frac{(16/3) \log n}{32 \log n} = \frac{1}{6}.$$

On the event $T \leq m$, extend the sequence of steps to length m by adding dummy steps with $X_t = 1$ for $t > T$; this can only increase S_m relative to S_T and thus can only help success. Hence,

$$\Pr[\text{failure}] \leq \Pr[T > m] + \Pr\left[2S_m - m < \log n\right].$$

It remains to bound $\Pr[2S_m - m < \log n]$. For $t \leq m$, we have $\mathbb{E}[X_t \mid \mathcal{F}_{t-1}] \geq 7/8$, so $\mathbb{E}[S_m] \geq (7/8)m = 28 \log n$. Let $\delta \in (0, 1)$ be defined by requiring

$$(1 - \delta)\mathbb{E}[S_m] = \frac{m + \log n}{2} = \frac{32 \log n + \log n}{2} = \frac{33}{2} \log n.$$

Since $\mathbb{E}[S_m] \geq 28 \log n$, we may take

$$1 - \delta \leq \frac{(33/2) \log n}{28 \log n} = \frac{33}{56},$$

so $\delta \geq 23/56$. By the Chernoff-Hoeffding bound for (conditionally) independent Bernoulli trials with lower bounded means, we have

$$\Pr\left[S_m \leq (1 - \delta)\mathbb{E}[S_m]\right] \leq \exp\left(-\frac{\delta^2}{2}\mathbb{E}[S_m]\right) \leq \exp\left(-\frac{(23/56)^2}{2} \cdot 28 \log n\right) = n^{-\Omega(1)}.$$

Consequently,

$$\Pr\left[2S_m - m < \log n\right] \leq n^{-\Omega(1)}.$$

Combining with the bound on $\Pr[T > m]$ yields

$$\Pr[\text{failure}] \leq \frac{1}{6} + n^{-\Omega(1)}.$$

By increasing the constant in $m = c \log n$ and reducing p by choosing a larger constant t (still $O(1)$), we can make the drift larger and the Chernoff exponent larger; in particular for any fixed constant k one can choose constants so that $\Pr[\text{failure}] \leq n^{-k}$. Thus the protocol succeeds with high probability on every input, and the expected number of communicated bits is $O(\log n)$.

This completes the proof. □

D.1.3 Claude

Proof. We provide a complete proof of both parts of the lemma.

Part 1: Correctness condition.

Let i^* denote the smallest index where $x_{i^*} \neq y_{i^*}$. We first establish that the protocol succeeds if the number of correct choices exceeds the number of wrong choices by $\log n$.

Consider the path in the extended tree from the root to the leaf corresponding to coordinate i^*. This path has length exactly $\log n + 3\log n = 4\log n$, since the binary tree has depth $\log n$ and each leaf has an attached path of length $3\log n$.

At each vertex v on this true path, a correct choice means either: (i) the hash correctly confirms that the first difference lies in I_v and the protocol moves to the correct child, or (ii) the hash correctly rejects I_v when $i^* \notin I_v$ and the protocol moves to the parent. A wrong choice means the hash gives an incorrect answer, causing the protocol to move away from the true path.

Let C denote the number of correct choices and W denote the number of wrong choices made by the protocol. We track the protocol's position using a potential function Φ defined as the distance from the current vertex to the nearest vertex on the true path to i^*.

Initially, $\Phi = 0$ since we start at the root, which is on the true path. Each correct choice either maintains $\Phi = 0$ (if we are on the true path and move along it) or decreases Φ by 1 (if we move toward the true path). Each wrong choice increases Φ by at most 1.

After C correct choices and W wrong choices, we have $\Phi \leq W - C'$ where C' is the number of correct choices that decreased Φ. Since each correct choice either decreases Φ or keeps us on the true path, if $C - W \geq \log n$, then even in the worst case where all wrong choices move us away and we need to recover, we can reach the leaf at depth $\log n$ on the true path.

More precisely, to reach the leaf corresponding to i^* in the binary tree portion (depth $\log n$), we need the net progress toward this leaf to be at least $\log n$. If $C - W \geq \log n$, then the net downward movement is sufficient to reach depth $\log n$ and identify the unique coordinate i^*. Once at this leaf, the attached path of length $3\log n$ provides additional confirmation steps, but the coordinate is already determined.

Therefore, if $C - W \geq \log n$, the protocol successfully computes i^*.

Part 2: Success probability via Chernoff-Hoeffding bound.

At each vertex, the parties use hash functions from a universal family. For a properly chosen universal hash family with range $\{0,1\}^c$ for constant c, the probability that a hash gives the correct answer is at least $1 - 2^{-c} \geq 1 - \delta$ for some small constant $\delta < 1/2$ (by choosing c large enough).

The protocol terminates after at most $T = 4\log n$ steps, since the tree has depth $4\log n$ and the protocol makes at most one move per step. Let X_j be the indicator that the j-th hash gives the correct answer. Then X_j are independent Bernoulli random variables with $\mathbb{E}[X_j] \geq p = 1 - \delta$ for constant $\delta < 1/2$.

Let $C = \sum_{j=1}^{T} X_j$ be the total number of correct choices and $W = T - C$ be the number of wrong choices. We need $C - W = 2C - T \geq \log n$.

This is equivalent to $C \geq \frac{T + \log n}{2} = \frac{4\log n + \log n}{2} = \frac{5\log n}{2}$.

The expected number of correct choices is $\mathbb{E}[C] = Tp \geq 4\log n \cdot (1 - \delta)$. For $\delta < 1/8$, we have $\mathbb{E}[C] \geq \frac{7\log n}{2} > \frac{5\log n}{2}$.

By the Chernoff-Hoeffding bound, for $C = \sum_{j=1}^{T} X_j$ with $X_j \in \{0,1\}$ independent,

$$\mathbb{P}\left[C < \mathbb{E}[C] - t\right] \leq \exp\left(-\frac{2t^2}{T}\right).$$

Setting $t = \mathbb{E}[C] - \frac{5\log n}{2} \geq \log n$ (for appropriate choice of δ), we get

$$\mathbb{P}\left[C < \frac{5\log n}{2}\right] \leq \exp\left(-\frac{2(\log n)^2}{4\log n}\right) = \exp\left(-\frac{\log n}{2}\right) = n^{-1/2}.$$

Therefore, the protocol succeeds with probability at least $1 - n^{-1/2}$, which is high probability. The total communication is $O(T) = O(\log n)$ bits since at each of the $O(\log n)$ vertices visited, the parties exchange $O(1)$ hash bits. □

D.1.4 Gemini

Proof. Let $x, y \in \{0,1\}^n$ be the input strings with $x \neq y$. Let i^* be the smallest index such that $x_{i^*} \neq y_{i^*}$. The protocol operates on the tree structure $\mathcal{T}$ described in the lemma. Let P^* denote the unique path from the root to the leaf corresponding to the interval $\{i^*\}$, extended by the path of length $3\log n$ attached to that leaf. We refer to P^* as the "correct path". The nodes on P^* are the "correct nodes".

We define a potential function $\Phi : V(\mathcal{T}) \to \mathbb{Z}$ for every node v in the tree. If $v \in P^*$, let $\Phi(v) = \text{depth}(v)$, where the root is at depth 0. If $v \notin P^*$, let u be the lowest ancestor of v that lies on P^*. We define $\Phi(v) = \text{depth}(u) - \text{dist}(v, u)$, where $\text{dist}(v, u)$ is the distance between v and u in the tree. Note that for the leaf node corresponding to i^*, the potential is $\log n$. For any node v in the extension path attached to the leaf for i^*, $\Phi(v) \geq \log n$.

We analyze the change in potential $\Delta\Phi$ in each step of the protocol. Let v_t be the node at step t. First, consider the case where $v_t \in P^*$. The interval I_{v_t} contains i^*. If the protocol makes the correct choice, it verifies $i^* \in I_{v_t}$ and moves to the child of v_t that lies on P^*. In this case, $\Phi(v_{t+1}) = \text{depth}(v_t) + 1 = \Phi(v_t) + 1$. If the protocol makes a wrong choice, it either moves to the parent of v_t (decreasing depth by 1) or moves to a child not on P^* (increasing distance from P^* by 1). In either case, $\Phi(v_{t+1}) = \Phi(v_t) - 1$.

Next, consider the case where $v_t \notin P^*$. The interval I_{v_t} does not contain i^*. The correct choice is for the hash check to reveal $i^* \notin I_{v_t}$, causing the protocol to move to the parent of v_t. Let u be the lowest ancestor of v_t on P^*. The parent of v_t is one step closer to u. Thus, $\Phi(v_{t+1}) = \text{depth}(u) - (\text{dist}(v_t, u) - 1) = \Phi(v_t) + 1$. If the protocol makes a wrong choice, it moves to a child of v_t, increasing the distance from u. Thus, $\Phi(v_{t+1}) = \Phi(v_t) - 1$.

Let N_C be the number of correct choices and N_W be the number of wrong choices after T steps. Since the protocol starts at the root where $\Phi(\text{root}) = 0$, the potential at step T is $\Phi(v_T) = N_C - N_W$. The lemma requires proving that if $N_C - N_W \geq \log n$, the protocol succeeds. If $\Phi(v_T) \geq \log n$, then by the definition of Φ, v_T must be a node with depth at least $\log n$ on the correct path P^*. The only nodes in $\mathcal{T}$ with $\Phi(v) \geq \log n$ are the leaf for i^* and the nodes on its extension path. All such nodes are associated with the correct index i^*. Thus, if $N_C - N_W \geq \log n$, the protocol correctly identifies i^*.

Now we analyze the probability of success and the communication complexity. In each step, the parties exchange $O(1)$ bits to perform hash comparisons. By using a sufficient constant number of bits (e.g., repeating the hash with independent seeds), we can ensure that the probability of a hash collision (a wrong choice) is bounded by a small constant $\epsilon < 1/2$. Let X_j be an indicator random variable for the j-th step, where $X_j = 1$ if the choice is correct and $X_j = -1$ if the choice is wrong. We have $\mathbb{P}(X_j = 1) \geq 1 - \epsilon$ and $\mathbb{P}(X_j = -1) \leq \epsilon$. Let $Y_j = (X_j + 1)/2$ be a Bernoulli variable with success probability $p \geq 1 - \epsilon$. The condition $N_C - N_W \geq \log n$ is equivalent to $\sum_{j=1}^{T} X_j \geq \log n$. Let the protocol run for $T = c \log n$ steps for a constant c to be determined. The total communication is $T \cdot O(1) = O(\log n)$ bits.

The expected net progress per step is $\mathbb{E}[X_j] \geq (1 - \epsilon) - \epsilon = 1 - 2\epsilon$. Let $\mu = 1 - 2\epsilon$. We choose ϵ such that $\mu > 0$ (e.g., $\epsilon = 1/4 \implies \mu = 1/2$). The expected total progress is $\mathbb{E}[\sum X_j] \geq \mu T = \mu c \log n$. We require $\sum X_j \geq \log n$. Using the Hoeffding bound for the sum $S_T = \sum_{j=1}^{T} X_j$ of independent bounded random variables in $[-1, 1]$:

$$\mathbb{P}(S_T \leq \mathbb{E}[S_T] - t) \leq \exp\left(-\frac{t^2}{2T}\right)$$

We want to bound $\mathbb{P}(S_T < \log n)$. Let $t = \mathbb{E}[S_T] - \log n \geq (\mu c - 1)\log n$. For this to be a valid bound, we need $\mu c > 1$. Let us choose $c = \frac{2}{\mu} + 1$. Then $t \geq 2\log n$.

$$\mathbb{P}(S_T < \log n) \leq \exp\left(-\frac{(2\log n)^2}{2c\log n}\right) = \exp\left(-\frac{2}{c}\log n\right) = n^{-2/c}$$

By choosing c sufficiently large (which corresponds to running the protocol for $O(\log n)$ steps), we can make the failure probability bounded by n^{-k} for any desired constant k. Thus, the protocol succeeds with high probability, and the expected number of bits communicated is $O(\log n)$. □

D.2 Problem 3.2

D.2.1 Problem Statement

Theorem D.2. *There exists a protocol for computing greater-than with communication complexity* $\lceil \log(1/\epsilon) \rceil$ *such that if the inputs are sampled uniformly and independently, then the average case error of the protocol is at most* ϵ*.*

D.2.2 GPT

Proof. We are tasked with proving that there exists a protocol for computing the greater-than function with communication complexity $\lceil \log(1/\epsilon) \rceil$ such that if the inputs are sampled uniformly and independently, then the average case error of the protocol is at most ϵ.

Let the two parties involved in the communication be Alice and Bob. Alice holds a number x and Bob holds a number y. The goal of the protocol is for Alice and Bob to compute whether $x > y$, while minimizing the communication complexity and ensuring that the error probability is at most ϵ in the average case.

We first note that the total number of possible inputs to the problem is $n \times n$ where n is the size of the input domain. Without loss of generality, assume that $x, y \in \{1, 2, \ldots, n\}$, where n is large and that the inputs are chosen uniformly at random. Hence, the total number of possible input pairs (x, y) is n^2.

Step 1: Defining the Communication Protocol We construct a communication protocol based on randomization. The protocol operates as follows:

1. Alice sends a message to Bob that encodes a random partition of the input space $\{1, 2, \ldots, n\}$ into roughly $1/\epsilon$ subsets. 2. Bob then receives this message, and based on the partition, he computes an approximation of whether $x > y$.

In the case of this protocol, the randomness ensures that the error rate of the protocol is controlled in expectation.

Step 2: Bounding the Error Probability The key to ensuring that the error probability is at most ϵ lies in the analysis of the randomization. Since the inputs are uniformly distributed, the probability of any particular input being incorrectly classified is directly related to the number of subsets into which the space is divided. If the partition is sufficiently fine (i.e., there are $\lceil \log(1/\epsilon) \rceil$ subsets), then the probability of misclassification on any given input pair (x, y) is at most ϵ.

Step 3: Communication Complexity The communication complexity of this protocol is determined by the number of subsets in the partition, which is proportional to $\lceil \log(1/\epsilon) \rceil$. To see this, consider that for each subset, Alice needs to communicate enough information to uniquely identify which subset her input belongs to. Since the total number of subsets is at most $1/\epsilon$, the number of bits Alice needs to communicate is at most $\lceil \log(1/\epsilon) \rceil$.

Step 4: Average Case Error Analysis Since the inputs are chosen uniformly and independently, the expected error probability over all input pairs is at most ϵ. This is because the probability of Alice and Bob miscommunicating depends on the partitioning of the input space, and as the number of subsets increases (with $\lceil \log(1/\epsilon) \rceil$ subsets), the probability of error decreases to the desired level of ϵ.

Thus, by ensuring the partition is sufficiently fine, we achieve the desired communication complexity and error bound.

Conclusion We have constructed a communication protocol with communication complexity $\lceil \log(1/\epsilon) \rceil$ that ensures an average case error probability of at most ϵ, as required by the theorem. □

D.2.3 Claude

Proof. Let $k = \lceil \log(1/\epsilon) \rceil$. We construct a protocol Π for computing the greater-than function with communication complexity k bits.

Let $x, y \in \{0, 1, \ldots, n-1\}$ be the inputs held by Alice and Bob respectively, where n is sufficiently large. The protocol proceeds as follows:

Alice partitions the domain $\{0, 1, \ldots, n-1\}$ into 2^k intervals of equal size. Specifically, for $i = 0, 1, \ldots, 2^k - 1$, define the interval $I_i = [i \cdot \lfloor n/2^k \rfloor, (i+1) \cdot \lfloor n/2^k \rfloor)$, with the last interval extended to include $n - 1$.

Alice sends Bob a k-bit message encoding the index i such that $x \in I_i$. Bob compares his input y with the intervals and outputs:

- 1 (meaning $x > y$) if y is in an interval I_j with $j < i$
- 0 (meaning $x \leq y$) if y is in an interval I_j with $j > i$
- An arbitrary value if y is in the same interval I_i as x

The communication complexity is exactly $k = \lceil \log(1/\epsilon) \rceil$ bits.

Now we analyze the average case error when x and y are sampled uniformly and independently from $\{0, 1, \ldots, n-1\}$.

The protocol errs only when x and y fall in the same interval I_i for some i. For each interval I_i of size s_i, the probability that both x and y fall in I_i is $(s_i/n)^2$.

Since each interval has size approximately $n/2^k$, we have $s_i \leq \lceil n/2^k \rceil$ for all i. The total error probability is:

$$\Pr[\text{error}] = \sum_{i=0}^{2^k-1} \Pr[x, y \in I_i] = \sum_{i=0}^{2^k-1} \frac{s_i^2}{n^2} \leq \sum_{i=0}^{2^k-1} \frac{s_i}{n} \cdot \frac{s_i}{n}$$

Since $\sum_{i=0}^{2^k-1} s_i = n$, and by the Cauchy-Schwarz inequality:

$$\sum_{i=0}^{2^k-1} s_i^2 \leq \left(\sum_{i=0}^{2^k-1} s_i \right) \cdot \max_i s_i \leq n \cdot \frac{n}{2^k} \cdot \left(1 + \frac{2^k}{n} \right)$$

For sufficiently large n, this gives:

$$\Pr[\text{error}] \leq \frac{1}{2^k} + O(1/n)$$

Since $k = \lceil \log(1/\epsilon) \rceil$, we have $2^k \geq 2^{\log(1/\epsilon)} = 1/\epsilon$, which implies:

$$\Pr[\text{error}] \leq \frac{1}{2^k} \leq \frac{1}{1/\epsilon} = \epsilon$$

Therefore, the protocol achieves average case error at most ϵ with communication complexity $\lceil \log(1/\epsilon) \rceil$. □

D.2.4 Gemini

Proof. Let n be the length of the input strings $x, y \in \{0,1\}^n$. We interpret x and y as integers in the range $[0, 2^n - 1]$ via the standard binary representation. The function to be computed is $GT_n(x, y)$, which is 1 if $x > y$ and 0 otherwise. The inputs x and y are sampled uniformly and independently from $\{0,1\}^n$.

Let $k = \lceil \log(1/\epsilon) \rceil$. We distinguish two cases based on the relationship between n and k.

Case 1: $n \leq k$. In this case, the protocol instructs Alice to send her entire input x to Bob. The communication cost is n bits, which satisfies $n \leq k = \lceil \log(1/\epsilon) \rceil$. Bob, possessing both x and y, can compute $GT_n(x, y)$ exactly. The error probability is 0, which is strictly less than ϵ.

Case 2: $n > k$. We define the protocol Π as follows:

1. Alice sends the prefix of x of length k, denoted by $x^{(k)} = x_1 x_2 \ldots x_k$, to Bob.
2. Bob compares $x^{(k)}$ with his own prefix $y^{(k)} = y_1 y_2 \ldots y_k$.
3. Bob outputs 1 if $x^{(k)} > y^{(k)}$ (interpreted as integers).
4. Bob outputs 0 otherwise (i.e., if $x^{(k)} \leq y^{(k)}$).

The communication complexity of this protocol is exactly $k = \lceil \log(1/\epsilon) \rceil$ bits.

We now analyze the average case error. Let $\Pi(x, y)$ denote the output of the protocol. The error event $\mathcal{E}$ occurs when $\Pi(x, y) \neq GT_n(x, y)$. We analyze the output based on the lexicographical comparison of the prefixes:

1. If $x^{(k)} > y^{(k)}$, then for any suffixes $x_{k+1} \dots x_n$ and $y_{k+1} \dots y_n$, it holds that $x > y$. In this case, $\Pi(x, y) = 1$ and $GT_n(x, y) = 1$. Thus, no error occurs. 2. If $x^{(k)} < y^{(k)}$, then $x < y$. In this case, $\Pi(x, y) = 0$ and $GT_n(x, y) = 0$. Thus, no error occurs. 3. If $x^{(k)} = y^{(k)}$, the protocol outputs 0. An error occurs if and only if $x > y$.

Therefore, the error event is a subset of the event that the prefixes are equal:

$$\mathcal{E} \subseteq \{(x, y) \mid x^{(k)} = y^{(k)}\}.$$

Consequently, the probability of error is bounded by the probability that the k-bit prefixes are identical:

$$\Pr[\text{error}] \leq \Pr_{x,y \sim \mathcal{U}(\{0,1\}^n)}[x^{(k)} = y^{(k)}].$$

Since x and y are chosen uniformly and independently from $\{0, 1\}^n$, their prefixes $x^{(k)}$ and $y^{(k)}$ are distributed uniformly and independently over $\{0, 1\}^k$. The probability that two independent, uniformly distributed k-bit strings are equal is:

$$\Pr[x^{(k)} = y^{(k)}] = \sum_{z \in \{0,1\}^k} \Pr[x^{(k)} = z] \cdot \Pr[y^{(k)} = z] = \sum_{z \in \{0,1\}^k} \left(\frac{1}{2^k}\right)^2 = 2^k \cdot \frac{1}{2^{2k}} = \frac{1}{2^k}.$$

Substituting the value of k:

$$\Pr[\text{error}] \leq \frac{1}{2^{\lceil \log(1/\epsilon) \rceil}} \leq \frac{1}{2^{\log(1/\epsilon)}} = \frac{1}{1/\epsilon} = \epsilon.$$

Thus, the protocol has communication complexity $\lceil \log(1/\epsilon) \rceil$ and an average case error of at most ϵ. □

D.3 Problem 3.3

D.3.1 Problem Statement

Theorem D.3. *In this exercise, we explore the concept of locality sensitive hashing. Suppose Alice and Bob are given inputs $x, y \in \{0, 1\}^n$ and they want to know whether the two strings are close in Hamming distance, or far apart. Show that for every $t < n/2$, there is a public-coin protocol of length $O(1)$ with the property that if the Hamming distance between x, y is at most t, the protocol outputs 1 with probability $2/3$, and if the Hamming distance is at least $2t$, the protocol outputs 0 with probability at least $2/3$.*

Hint: Consider taking the inner product of x, y with random strings z generated using a biased coin, and compute $\mathbb{E}\left[(-1)^{\langle x-y,z \rangle}\right]$.

D.3.2 GPT

Proof. Let $x, y \in \{0, 1\}^n$ be two binary strings. We wish to construct a public-coin protocol to determine whether the Hamming distance between x and y is small or large. Specifically, we want the protocol to satisfy the following:

- If the Hamming distance between x and y is at most t, the protocol outputs 1 with probability at least $2/3$. - If the Hamming distance between x and y is at least $2t$, the protocol outputs 0 with probability at least $2/3$.

To achieve this, we will use the inner product of x and y with random binary strings $\zeta \in \{0, 1\}^n$, where ζ is generated using a biased coin. The protocol computes the expectation of the value $(-1)^{\langle x-y,\zeta \rangle}$.

Let $\zeta = (\zeta_1, \dots, \zeta_n)$ be a random binary string, where each ζ_i is independently chosen to be 1 with probability p and 0 with probability $1 - p$, for some p.

The value we are interested in is the expectation $\mathbb{E}\left[(-1)^{\langle x-y,\zeta \rangle}\right]$, where $\langle x - y, \zeta \rangle$ denotes the inner product of the binary vectors $x - y$ and ζ. This inner product is given by

$$\langle x - y, \zeta \rangle = \sum_{i=1}^{n} (x_i - y_i)\zeta_i.$$

Now, let us analyze the behavior of this expectation depending on the Hamming distance between x and y. The Hamming distance between x and y is given by

$$d_H(x, y) = \sum_{i=1}^{n} \mathbf{1}_{\{x_i \neq y_i\}},$$

which counts the number of positions where x and y differ.

1. **Case 1: The Hamming distance is at most t.**

In this case, we have $d_H(x, y) \leq t$. The expectation of the inner product is computed as

$$\mathbb{E}\left[(-1)^{\langle x-y,\zeta\rangle}\right] = \mathbb{E}\left[\prod_{i=1}^{n}(-1)^{(x_i-y_i)\zeta_i}\right].$$

Since ζ_i is independent of x_i and y_i, we can express this expectation as the product of expectations over each i:

$$\prod_{i=1}^{n} \mathbb{E}\left[(-1)^{(x_i-y_i)\zeta_i}\right].$$

For each i, we have two cases depending on whether $x_i = y_i$ or $x_i \neq y_i$. If $x_i = y_i$, the contribution to the expectation is 1, since $(-1)^{(x_i-y_i)\zeta_i} = 1$ for all ζ_i. If $x_i \neq y_i$, then the contribution is

$$\mathbb{E}\left[(-1)^{\zeta_i}\right] = p \cdot (-1) + (1 - p) \cdot 1 = 1 - 2p.$$

Thus, the overall expectation is

$$\prod_{i:x_i \neq y_i} (1 - 2p).$$

Since the number of positions where $x_i \neq y_i$ is at most t, we have

$$\mathbb{E}\left[(-1)^{\langle x-y,\zeta\rangle}\right] = (1 - 2p)^t.$$

We want this expectation to be at least $2/3$, so we require

$$(1 - 2p)^t \geq \frac{2}{3}.$$

For sufficiently small p, we can ensure this condition holds. Specifically, if p is chosen such that

$$1 - 2p \approx 1 - \frac{1}{t},$$

then the expectation will be at least $2/3$.

2. **Case 2: The Hamming distance is at least $2t$.**

In this case, we have $d_H(x, y) \geq 2t$. The expectation is again given by

$$\mathbb{E}\left[(-1)^{\langle x-y,\zeta\rangle}\right] = \prod_{i:x_i \neq y_i} (1 - 2p).$$

Since $d_H(x, y) \geq 2t$, the number of terms in the product is at least $2t$. Thus, we have

$$\mathbb{E}\left[(-1)^{\langle x-y,\zeta\rangle}\right] = (1 - 2p)^{2t}.$$

We want this expectation to be at most $1/3$, so we require

$$(1-2p)^{2t} \leq \frac{1}{3}.$$

For sufficiently small p, we can ensure this condition holds. Specifically, if

$$1-2p \approx 1-\frac{1}{2t},$$

then the expectation will be at most $1/3$.

Thus, we have constructed a public-coin protocol that satisfies the desired properties. The protocol uses a constant-length interaction (of length $O(1)$), and the probability of correctly determining whether the Hamming distance is small or large is at least $2/3$ in both cases. Therefore, the lemma holds. □

D.3.3 Claude

Proof. We construct a public-coin protocol that uses locality sensitive hashing based on random projections.

Let $p = \frac{1}{2} - \frac{1}{4t}$ where $t < n/2$. The protocol proceeds as follows: Alice and Bob publicly generate a random string $z \in \{0,1\}^n$ where each bit z_i is independently set to 1 with probability p and 0 with probability $1-p$. Alice computes $a = \langle x, z\rangle \bmod 2$ and Bob computes $b = \langle y, z\rangle \bmod 2$. The protocol outputs 1 if $a = b$ and outputs 0 if $a \neq b$.

The communication complexity is $O(1)$ since Alice sends one bit a to Bob, who then compares it with his bit b.

We now analyze the success probability. Note that $a = b$ if and only if $\langle x-y, z\rangle \equiv 0 \pmod 2$. Let $d = d_H(x,y)$ denote the Hamming distance between x and y. Then $\langle x-y, z\rangle \equiv 0 \pmod 2$ if and only if an even number of the d positions where x and y differ have $z_i = 1$.

The probability that the protocol outputs 1 is:

$$\Pr[a=b] = \Pr[\langle x-y, z\rangle \equiv 0 \pmod 2] = \sum_{k \text{ even}} \binom{d}{k} p^k (1-p)^{d-k}$$

Using the hint, we compute:

$$\mathbb{E}[(-1)^{\langle x-y,z\rangle}] = \sum_{k=0}^{d} \binom{d}{k} p^k (1-p)^{d-k} (-1)^k = (1-2p)^d$$

Since $\mathbb{E}[(-1)^{\langle x-y,z\rangle}] = \Pr[\langle x-y,z\rangle \text{ even}] - \Pr[\langle x-y,z\rangle \text{ odd}]$ and these probabilities sum to 1, we have:

$$\Pr[a=b] = \frac{1+(1-2p)^d}{2}$$

Substituting $p = \frac{1}{2} - \frac{1}{4t}$, we get $1-2p = \frac{1}{2t}$, so:

$$\Pr[a=b] = \frac{1+(1/(2t))^d}{2}$$

Case 1: If $d \leq t$, then $(1/(2t))^d \geq (1/(2t))^t$. For $t < n/2$, we have $(1/(2t))^t \geq 1/3$ for sufficiently small t, giving $\Pr[a=b] \geq \frac{1+1/3}{2} = \frac{2}{3}$.

More precisely, since $d \leq t$, we have $(1/(2t))^d \geq (1/(2t))^t$. For the protocol to work, we need $(1/(2t))^t \geq 1/3$, which holds when $t \leq c$ for some constant c. For larger t, we can repeat the protocol $O(1)$ times independently and take the majority vote, maintaining $O(1)$ communication.

Case 2: If $d \geq 2t$, then $(1/(2t))^d \leq (1/(2t))^{2t} = (1/(4t^2))^t$. For any fixed $t \geq 1$, we have $(1/(4t^2))^t \to 0$ rapidly, so $\Pr[a=b] \leq \frac{1+(1/(4t^2))^t}{2} \leq \frac{1}{3}$ for t sufficiently large. Thus $\Pr[a \neq b] \geq \frac{2}{3}$.

By adjusting constants and using a constant number of repetitions with majority voting, we obtain a protocol with the desired properties: communication complexity $O(1)$, success probability at least $2/3$ when $d \leq t$, and success probability at least $2/3$ when $d \geq 2t$. □

D.3.4 Gemini

Proof. Let $x, y \in \{0,1\}^n$ and let $d = d_H(x, y)$ denote the Hamming distance between them. We construct a public-coin protocol using the hint provided.

Consider a random string $z \in \{0,1\}^n$ where each bit z_i is chosen independently from a Bernoulli distribution with parameter $\rho \in (0, 1/2)$. That is, $\Pr[z_i = 1] = \rho$ and $\Pr[z_i = 0] = 1 - \rho$. We analyze the random variable $V = (-1)^{\langle x-y,z\rangle}$. Note that the inner product in the exponent is over the integers, but since we are interested in the value modulo 2, we observe that $\langle x - y, z\rangle \equiv \sum_{i=1}^{n}(x_i - y_i)z_i \pmod 2$. The term $(x_i - y_i)$ is non-zero modulo 2 if and only if $x_i \neq y_i$. Let $S = \{i \mid x_i \neq y_i\}$ be the set of indices where x and y differ, so $|S| = d$. Then $\langle x - y, z\rangle \equiv \sum_{i\in S} z_i \pmod 2$.

We compute the expectation of V:

$$\mathbb{E}[V] = \mathbb{E}\left[(-1)^{\sum_{i\in S} z_i}\right] = \mathbb{E}\left[\prod_{i\in S}(-1)^{z_i}\right].$$

Since the bits z_i are independent, the expectation of the product is the product of the expectations. For any i,

$$\mathbb{E}[(-1)^{z_i}] = (1-\rho)(-1)^0 + \rho(-1)^1 = 1 - 2\rho.$$

Therefore,

$$\mathbb{E}[V] = \prod_{i\in S}(1 - 2\rho) = (1 - 2\rho)^d.$$

Let A be the event that $\langle x, z\rangle \equiv \langle y, z\rangle \pmod 2$. This is equivalent to $\langle x - y, z\rangle \equiv 0 \pmod 2$, which corresponds to $V = 1$. If the inner products differ, $V = -1$. Let $p(d) = \Pr[A]$. We can express the expectation as:

$$\mathbb{E}[V] = 1 \cdot \Pr[A] + (-1) \cdot \Pr[A^c] = p(d) - (1 - p(d)) = 2p(d) - 1.$$

Solving for $p(d)$, we obtain the probability that the hash values agree:

$$p(d) = \frac{1 + \mathbb{E}[V]}{2} = \frac{1 + (1 - 2\rho)^d}{2}.$$

We now specify the protocol. Alice and Bob use public randomness to generate k independent strings $z^{(1)}, \ldots, z^{(k)}$ using the distribution described above with a specific parameter ρ. Alice computes the bits $a_j = \langle x, z^{(j)}\rangle \pmod 2$ for $j = 1, \ldots, k$ and sends the vector $a = (a_1, \ldots, a_k)$ to Bob. Bob computes $b_j = \langle y, z^{(j)}\rangle \pmod 2$ for $j = 1, \ldots, k$. Bob calculates the fraction of indices j where $a_j = b_j$. If this fraction is strictly greater than a threshold τ, Bob outputs 1; otherwise, he outputs 0. The communication cost is k bits.

We choose ρ such that $(1 - 2\rho)^t = 1/2$. This requires $1 - 2\rho = (1/2)^{1/t}$. Since $t \geq 1$, we have $1/2 \leq (1/2)^{1/t} < 1$, which implies a valid $\rho \in (0, 1/4]$. With this choice, we analyze the collision probability $p(d)$ for the two cases.

Case 1: The Hamming distance is small, $d \leq t$. Since $0 < 1 - 2\rho < 1$, the term $(1 - 2\rho)^d$ is monotonically decreasing in d. Thus,

$$p(d) \geq p(t) = \frac{1 + (1 - 2\rho)^t}{2} = \frac{1 + 1/2}{2} = \frac{3}{4}.$$

Case 2: The Hamming distance is large, $d \geq 2t$.

$$p(d) \leq p(2t) = \frac{1 + (1 - 2\rho)^{2t}}{2} = \frac{1 + ((1 - 2\rho)^t)^2}{2} = \frac{1 + (1/2)^2}{2} = \frac{1 + 1/4}{2} = \frac{5}{8}.$$

We set the threshold $\tau = \frac{3/4+5/8}{2} = \frac{11}{16}$. Let Y_j be the indicator variable for the event $a_j = b_j$. The variables Y_j are independent Bernoulli trials with success probability $p(d)$. Let $\bar{Y} = \frac{1}{k}\sum_{j=1}^{k} Y_j$ be the fraction of agreements.

If $d \leq t$, $\mathbb{E}[\bar{Y}] \geq 3/4$. The protocol fails if $\bar{Y} \leq 11/16$. By the Hoeffding inequality, the probability of error is:

$$\Pr[\bar{Y} \leq 11/16] \leq \Pr[\bar{Y} \leq \mathbb{E}[\bar{Y}] - 1/16] \leq e^{-2k(1/16)^2} = e^{-k/128}.$$

If $d \geq 2t$, $\mathbb{E}[\bar{Y}] \leq 5/8$. The protocol fails if $\bar{Y} > 11/16$. By the Hoeffding inequality, the probability of error is:

$$\Pr[\bar{Y} > 11/16] \leq \Pr[\bar{Y} \geq \mathbb{E}[\bar{Y}] + 1/16] \leq e^{-2k(1/16)^2} = e^{-k/128}.$$

We require the error probability to be at most $1/3$. Setting $e^{-k/128} \leq 1/3$, we find that $k \geq 128 \ln 3$ suffices. Since $128 \ln 3$ is a constant independent of n, we can choose a constant k (e.g., $k = 141$) to satisfy the requirements. Thus, the protocol has length $O(1)$ and achieves the desired success probabilities. □

D.4 Problem 3.4

D.4.1 Problem Statement

Theorem D.4. *Suppose Alice and Bob are given sets $X, Y \subseteq [n]$ with the promise that $|X| \leq k$, and Y is allowed to have arbitrary size. There exists a randomized protocol for the lopsided disjointness problem that determines whether the sets are disjoint with worst-case error $1/3$, such that in every execution of the protocol, Alice sends at most $O(k)$ bits and Bob sends at most $O(n)$ bits.*

D.4.2 GPT

Proof. We are tasked with proving the existence of a randomized protocol for the lopsided disjointness problem, where Alice and Bob are given sets $X, Y \subseteq [n]$ with the conditions that $|X| \leq k$, and Y can have arbitrary size. The protocol should determine whether the sets X and Y are disjoint with worst-case error at most $1/3$, such that Alice sends at most $O(k)$ bits, and Bob sends at most $O(n)$ bits in each execution.

We will construct a randomized protocol based on the following steps:

1. Alice's Strategy: Alice begins by selecting a random subset $S \subseteq X$ of size at most k, where each element of X is included in S independently with probability p. Specifically, Alice chooses each element of X to be in S with probability $p = \frac{k}{n}$. Since $|X| \leq k$, we have $|S| \leq k$ with high probability, and the expected size of S is kp.

Alice then sends the set S to Bob. Since $|S| \leq k$, the number of bits Alice sends is at most $O(k)$.

2. Bob's Strategy: Upon receiving S, Bob checks whether S and Y are disjoint. If $S \cap Y = \emptyset$, Bob outputs "disjoint". Otherwise, if $S \cap Y \neq \emptyset$, Bob outputs "not disjoint".

The correctness of this step follows directly from the fact that if $S \cap Y \neq \emptyset$, then $X \cap Y \neq \emptyset$. However, the protocol is randomized, and thus the probability of error needs to be considered.

3. Analysis of the Error Probability: We now compute the probability of error in this protocol. There are two possible cases:

- Case 1: X and Y are disjoint: If $X \cap Y = \emptyset$, then for any $S \subseteq X$, we have $S \cap Y = \emptyset$ with high probability. Specifically, the probability that S and Y are not disjoint is bounded by $|S|/n$, which is at most k/n. Hence, the probability that S and Y are disjoint is at least $1 - k/n$. Thus, the error probability in this case is at most k/n.

- Case 2: X and Y are not disjoint: If $X \cap Y \neq \emptyset$, then with probability at least $p = \frac{k}{n}$, the set S chosen by Alice will intersect Y. The probability that $S \cap Y = \emptyset$ is at most $(1-p)^k \approx e^{-k^2/n}$ by a standard union bound argument. Therefore, the error probability in this case is at most $e^{-k^2/n}$.

4. Error Bound: To ensure that the protocol succeeds with high probability, we require that the error probability be at most $1/3$. Hence, we set the error probability in both cases to be at most $1/3$, which implies that we must have $k/n \leq 1/3$ and $e^{-k^2/n} \leq 1/3$.

5. Communication Complexity: - Alice sends the set S to Bob, which requires at most $O(k)$ bits, since $|S| \leq k$. - Bob only needs to check whether $S \cap Y = \emptyset$, which can be done in $O(n)$ time, as it involves checking for each element of S whether it is in Y.

Thus, the communication complexity is $O(k)$ for Alice and $O(n)$ for Bob, and the error probability is bounded by $1/3$.

Therefore, the randomized protocol described above satisfies the conditions of the theorem. The proof is complete. □

D.4.3 Claude

Proof. We construct a randomized protocol for the lopsided disjointness problem where Alice holds $X \subseteq [n]$ with $|X| \leq k$ and Bob holds $Y \subseteq [n]$ of arbitrary size.

The protocol proceeds as follows. Let p be a prime with $n < p < 2n$, which exists by Bertrand's postulate. Let $\mathbb{F}_p$ denote the finite field of order p.

Alice and Bob publicly share a random hash function $h : [n] \to \mathbb{F}_p$ chosen uniformly at random from a pairwise independent family of hash functions. Such a family can be described by $h_a(x) = ax \bmod p$ for $a \in \mathbb{F}_p^*$ chosen uniformly at random.

Alice computes the set $H_X = \{h(x) : x \in X\}$ and sends this set to Bob. Since $|X| \leq k$ and each element of H_X requires $O(\log p) = O(\log n)$ bits to represent, Alice sends at most $O(k \log n)$ bits. By using a more efficient encoding, this can be reduced to $O(k)$ bits by sending only the differences or using appropriate data structures.

Bob checks whether there exists any $y \in Y$ such that $h(y) \in H_X$. If such a y exists, Bob outputs that the sets are not disjoint. Otherwise, Bob outputs that the sets are disjoint. Bob performs this computation locally without sending any message, so Bob sends 0 bits.

We now analyze the correctness of this protocol.

If $X \cap Y = \emptyset$, then for all $x \in X$ and $y \in Y$, we have $x \neq y$. Since h is injective on $[n]$ with high probability (as $p > n$), we have $h(x) \neq h(y)$ for all such pairs. Therefore, Bob will correctly output that the sets are disjoint with probability 1.

If $X \cap Y \neq \emptyset$, then there exists some element $z \in X \cap Y$. In this case, $h(z) \in H_X$ and Bob will find that $h(z) \in H_X$ when checking $z \in Y$, so Bob will correctly output that the sets are not disjoint with probability 1.

However, to achieve the stated bit complexity where Bob sends $O(n)$ bits, we modify the protocol as follows.

Instead, we use a protocol where Bob also participates in the communication. Alice sends a fingerprint of her set X using $O(k)$ bits, and Bob sends a fingerprint of his set Y using $O(n)$ bits. Specifically, they use polynomial fingerprinting.

Alice represents her set X by the polynomial $P_X(z) = \prod_{x \in X}(z - x)$ over $\mathbb{F}_p$. Bob represents his set Y by the polynomial $P_Y(z) = \prod_{y \in Y}(z - y)$ over $\mathbb{F}_p$.

They choose a random evaluation point $r \in \mathbb{F}_p$ uniformly at random. Alice sends $P_X(r)$ to Bob using $O(\log p) = O(\log n)$ bits, which is $O(k)$ bits when accounting for the degree. Bob computes $P_Y(r)$ and checks whether $\gcd(P_X(r), P_Y(r)) \neq 1$ by checking if there exists $y \in Y$ such that $P_X(y) = 0$ modulo the fingerprint.

More precisely, Bob checks whether there exists $y \in Y$ such that $r - y$ divides $P_X(r)$ in the fingerprint space. Bob sends his result using $O(1)$ bits.

By the Schwartz-Zippel lemma, if $X \cap Y = \emptyset$, the polynomials P_X and P_Y are coprime, and the probability of a false positive is at most $\frac{\deg(P_X)+\deg(P_Y)}{p} \leq \frac{k+n}{p} \leq \frac{2n}{n} = 2/n$ for sufficiently large n, which can be made at most $1/3$ by appropriate choice of parameters.

Therefore, the protocol achieves worst-case error probability at most $1/3$, with Alice sending $O(k)$ bits and Bob sending $O(n)$ bits in the worst case (when Bob needs to report his computation result or send information about Y). □

D.4.4 Gemini

Proof. Let $X, Y \subseteq [n]$ be the inputs given to Alice and Bob respectively, with the constraint $|X| \leq k$. Let $DISJ(X, Y)$ be the function that evaluates to 1 if $X \cap Y = \emptyset$ and 0 otherwise. We define a deterministic protocol Π that solves this problem, which serves as a randomized protocol with error probability $0 \leq 1/3$.

Let $y \in \{0, 1\}^n$ be the characteristic vector of Bob's set Y, defined such that $y_i = 1$ if $i \in Y$ and $y_i = 0$ otherwise.

Protocol Π:

1. Bob sends the entire characteristic vector y to Alice.

2. Alice receives y. Using her set X, she checks for an intersection by computing:

$$b = \bigwedge_{i \in X} (1 - y_i)$$

   If for all $i \in X$, $y_i = 0$, then $b = 1$ (disjoint). If there exists some $i \in X$ such that $y_i = 1$, then $b = 0$ (not disjoint).

3. Alice sends the bit b to Bob.

4. Both parties output b.

Communication Complexity:

- **Bob's Communication:** Bob sends the vector y, which has length n. The number of bits sent by Bob is n. Since $n \leq O(n)$, this satisfies the constraint.

- **Alice's Communication:** Alice sends a single bit b. The number of bits sent by Alice is 1. Since $k \geq 1$ (or treating $O(k)$ as bounded below by a constant for the protocol overhead), $1 \leq O(k)$.

Correctness: The protocol is deterministic.

- If $X \cap Y = \emptyset$, then for every element $x \in X$, we have $x \notin Y$. Consequently, $y_x = 0$ for all $x \in X$. Alice computes $b = 1$. The output is correct.

- If $X \cap Y \neq \emptyset$, there exists at least one element $z \in X$ such that $z \in Y$. Consequently, $y_z = 1$. Alice observes this and computes $b = 0$. The output is correct.

The worst-case error probability is 0, which satisfies the requirement of being at most $1/3$. Thus, the protocol Π satisfies all conditions of the theorem. □

D.5 Problem 3.5

D.5.1 Problem Statement

Theorem D.5. *For a real matrix M and $\epsilon > 0$ define the ϵ-approximate rank of M to be* $\mathrm{rank}_\epsilon(M) = \min\{\mathrm{rank}(A) : |A_{i,j} - M_{i,j}| \leq \epsilon \text{ for all } i, j\}$.

1. Find a Boolean matrix with rank r and $1/3$-approximate rank at most $O(\log r)$. Hint: The equality function.

2. More generally, suppose there is a private coin protocol computing a Boolean function with communication complexity c and error ϵ. Then show that $\mathrm{rank}_\epsilon(M) \leq 2^c$.

3. Prove the following strengthening of Theorem 2.17. The communication complexity of a matrix M is at most

$$O\left(\sqrt{\mathrm{rank}_{1/3}(M)} \log^2 \mathrm{rank}(M)\right).$$

D.5.2 GPT

Proof. **Part 1: Boolean matrix with rank r and $1/3$-approximate rank at most $O(\log r)$**

Let M be a Boolean matrix with rank r. Define the Boolean matrix M' to be the matrix where every entry $M'_{i,j}$ is the result of applying the equality function on $M_{i,j}$ with a tolerance of $\epsilon = 1/3$. Specifically, define

$$M'_{i,j} = \begin{cases} 1 & \text{if } M_{i,j} = 1, \\ 0 & \text{if } M_{i,j} = 0. \end{cases}$$

Since M' is a Boolean matrix, its entries are either 0 or 1, and the rank of M' is at most r because M' is simply a modification of M where only equality is tested. The ϵ-approximate rank of M' is the smallest

rank of any matrix A such that $|A_{i,j} - M'_{i,j}| \leq \epsilon = 1/3$ for all i, j. Since the matrix M' only consists of 0s and 1s, we conclude that for $\epsilon = 1/3$, the approximate rank is at most $O(\log r)$, where the $O(\log r)$ bound comes from the fact that the approximate rank depends logarithmically on the number of distinct entries (here, 2 distinct entries).

Thus, for any Boolean matrix of rank r, the $1/3$-approximate rank is at most $O(\log r)$.

Part 2: Private coin protocol and approximate rank bound

Let M be a matrix and suppose there exists a private coin protocol computing a Boolean function f with communication complexity c and error ϵ. We are tasked with proving that the ϵ-approximate rank of M satisfies the bound

$$\text{rank}_\epsilon(M) \leq 2^c.$$

Consider the private coin protocol that computes the Boolean function f. The communication complexity c implies that there exists a communication protocol between two parties where the total number of bits exchanged is at most c. Let M be the matrix associated with the communication problem, where each entry of the matrix corresponds to an input of the function f.

The error ϵ means that the function f computes an approximate solution with probability at least $1 - \epsilon$. Therefore, there exists an approximation of M with error at most ϵ in terms of the entries of the matrix. The fact that the communication complexity is c implies that the number of distinct possible communication patterns between the parties is at most 2^c. Each communication pattern can be interpreted as a distinct entry in the matrix, and since the total number of communication patterns is bounded by 2^c, the rank of the approximate matrix M_ϵ is bounded by 2^c.

Thus, we conclude that the ϵ-approximate rank of M is at most 2^c.

Part 3: Communication complexity bound for matrix M

We are given a matrix M and are tasked with proving that the communication complexity of M is at most

$$O\left(\sqrt{\text{rank}(M)/3}\log^2 \text{rank}(M)\right).$$

Let $\text{rank}(M) = r$. From the previous part, we know that the ϵ-approximate rank of M is bounded by 2^c, where c is the communication complexity. Therefore, the communication complexity c is at least logarithmic in the rank of M, i.e.,

$$c \geq \log \text{rank}(M) = \log r.$$

Now, we aim to show that the communication complexity is not only bounded by $\log r$ but is in fact bounded by a stronger expression. The key observation is that for any matrix M with rank r, there exists a communication protocol that simulates a matrix decomposition of M, for which the communication complexity is related to the square root of the rank of M. Specifically, it is known that the communication complexity of matrix multiplication and related tasks can be bounded by

$$O\left(\sqrt{r}\log^2 r\right),$$

where the factor $\sqrt{r}$ arises from the structure of the matrix and the logarithmic factors come from the size of the communication channels. This bound on the communication complexity can be interpreted as a direct consequence of the matrix's rank and the efficiency of private coin protocols used to approximate matrix entries.

Therefore, the communication complexity of matrix M is at most

$$O\left(\sqrt{\text{rank}(M)/3}\log^2 \text{rank}(M)\right).$$

□

D.5.3 Claude

Proof. We prove each part of the theorem in turn.

Part 1: We construct a Boolean matrix with rank r and $1/3$-approximate rank at most $O(\log r)$.

Consider the equality function EQ : $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ defined by $\text{EQ}(x,y) = 1$ if $x = y$ and $\text{EQ}(x,y) = 0$ otherwise. Let M be the $2^n \times 2^n$ communication matrix of EQ, where rows are indexed by $x \in \{0,1\}^n$ and columns by $y \in \{0,1\}^n$, with $M_{x,y} = \text{EQ}(x,y)$.

The matrix M is the identity matrix of size $2^n \times 2^n$, hence $\text{rank}(M) = 2^n$. Setting $r = 2^n$, we have $n = \log r$.

There exists a randomized protocol for EQ with error at most $1/3$ and communication complexity $O(n)$: Alice and Bob use a random hash function from a pairwise independent family mapping $\{0,1\}^n$ to $\{0,1\}^k$ where $k = O(n)$. Alice sends the hash of her input to Bob, who compares it with the hash of his input. This protocol has error at most $1/3$ for appropriate choice of k, and uses $O(n) = O(\log r)$ bits of communication.

By Part 2 (which we prove next), this implies $\text{rank}_{1/3}(M) \leq 2^{O(n)} = 2^{O(\log r)} = r^{O(1)}$. More precisely, using the protocol with communication $c = O(\log r)$, we get $\text{rank}_{1/3}(M) \leq 2^c = 2^{O(\log r)} = O(r^\alpha)$ for some constant α. However, a more careful analysis using the specific structure shows $\text{rank}_{1/3}(M) = O(\log r)$ directly.

Part 2: Suppose there is a private coin protocol computing a Boolean function with communication complexity c and error ϵ. We show that $\text{rank}_\epsilon(M) \leq 2^c$.

Let Π be a private coin protocol with communication complexity c and error at most ϵ. For each transcript t of length at most c, there are at most 2^c possible transcripts. Define a rectangle $R_t = A_t \times B_t$ where A_t is the set of Alice's inputs consistent with transcript t and B_t is the set of Bob's inputs consistent with transcript t.

For each transcript t, let v_t be the output value (0 or 1) that the protocol produces on transcript t. Define the matrix A by

$$A_{x,y} = \sum_{t:(x,y) \text{ produces } t \text{ with positive probability}} \Pr[\text{transcript is } t \mid x, y] \cdot v_t.$$

Since the protocol has error at most ϵ, we have $|A_{x,y} - M_{x,y}| \leq \epsilon$ for all x, y.

Each transcript t defines a rank-1 matrix M_t where $(M_t)_{x,y} = \mathbf{1}_{x \in A_t} \cdot \mathbf{1}_{y \in B_t} \cdot v_t$. The matrix A can be written as a non-negative combination of these rank-1 matrices, hence $\text{rank}(A) \leq 2^c$.

Therefore, $\text{rank}_\epsilon(M) \leq 2^c$.

Part 3: We prove that the communication complexity of a matrix M is at most $O\left(\sqrt{\text{rank}_{1/3}(M)} \log^2 \text{rank}(M)\right)$.

Let $r = \text{rank}(M)$ and $s = \text{rank}_{1/3}(M)$. By definition, there exists a matrix A with $\text{rank}(A) \leq s$ such that $|A_{i,j} - M_{i,j}| \leq 1/3$ for all i, j.

Since M is Boolean and $|A_{i,j} - M_{i,j}| \leq 1/3$, we have that A approximates M with error $1/3$.

By the standard rank-communication complexity relationship (using the log-rank conjecture approach), we can decompose A as $A = \sum_{k=1}^{s} u_k v_k^T$ where u_k and v_k are vectors.

Using a sampling-based protocol: Alice and Bob can use shared randomness to sample $O(\sqrt{s} \log r)$ coordinates. By communicating these sampled values (each requiring $O(\log r)$ bits), they can approximate the inner products sufficiently well to determine $M_{i,j}$ with high probability.

The total communication is $O(\sqrt{s} \log r \cdot \log r) = O(\sqrt{s} \log^2 r) = O\left(\sqrt{\text{rank}_{1/3}(M)} \log^2 \text{rank}(M)\right)$. □

D.5.4 Gemini

Proof. **Part 1.** Let M be the Inequality matrix of size $r \times r$, defined by $M_{i,j} = 1$ if $i \neq j$ and $M_{i,j} = 0$ if $i = j$. This is the complement of the Equality function matrix I_r. First, we observe that $M = J - I$, where J is the all-ones matrix and I is the identity matrix. The eigenvalues of J are r (with multiplicity 1) and 0 (with multiplicity $r - 1$). Thus, the eigenvalues of $M = J - I$ are $r - 1$ and -1 (with multiplicity $r - 1$). Since all eigenvalues are non-zero, $\text{rank}(M) = r$.

Next, we bound the approximate rank of M. Consider the private coin randomized communication protocol for the Inequality function where Alice and Bob possess inputs $i, j \in \{1, \ldots, r\}$. They use a shared random hash function $h : \{1, \ldots, r\} \to \{0, 1\}^k$ for a constant k. Alice sends $h(i)$ to Bob. Bob outputs 1 if $h(i) \neq h(j)$ and 0 otherwise. If $i = j$, $h(i) = h(j)$ always, so the output is 0 with probability 1. If $i \neq j$, $\Pr[h(i) = h(j)] = 2^{-k}$. By choosing k sufficiently large (e.g., $k = 2$), the error probability is at most $1/4 < 1/3$. The communication cost is $c = O(1)$ bits. By the result in Part 2 (proven below), $\mathrm{rank}_{1/3}(M) \leq 2^c = 2^{O(1)} = O(1)$. Since $O(1) \leq O(\log r)$, M satisfies the requirements.

Part 2. Let Π be a private coin protocol computing the function M with communication complexity c and error ϵ. Let $\mathcal{R}_A$ and $\mathcal{R}_B$ denote the private random strings of Alice and Bob, respectively. For any inputs x, y, let $P(x, y)$ denote the probability that the protocol outputs 1. Since the protocol has error ϵ, we have $|P(x, y) - M_{x,y}| \leq \epsilon$ for all x, y. Thus, P is an ϵ-approximation of M. We must show $\mathrm{rank}(P) \leq 2^c$.

The probability $P(x, y)$ can be expressed as the sum over all accepting transcripts τ of the probability that τ is generated given inputs x, y:

$$P(x, y) = \sum_{\tau \in \mathcal{T}_{accept}} \Pr[\Pi(x, y) \text{ generates } \tau].$$

Since the protocol uses private coins, the probability of generating a specific transcript τ factors into the probability that Alice sends her messages consistent with τ and Bob sends his messages consistent with τ. Specifically, for a transcript τ, let $p_\tau(x)$ be the probability Alice sends the messages in τ given input x (averaging over $\mathcal{R}_A$), and let $q_\tau(y)$ be the probability Bob sends the messages in τ given input y (averaging over $\mathcal{R}_B$). Then:

$$\Pr[\Pi(x, y) \text{ generates } \tau] = p_\tau(x) q_\tau(y).$$

Substituting this back into the expression for $P(x, y)$:

$$P(x, y) = \sum_{\tau \in \mathcal{T}_{accept}} p_\tau(x) q_\tau(y).$$

This equation expresses the matrix P as a sum of rank-1 matrices (outer products of vectors p_τ and q_τ). The number of terms in the sum is the number of accepting transcripts, which is bounded by the total number of transcripts 2^c. By the subadditivity of rank,

$$\mathrm{rank}(P) \leq \sum_{\tau \in \mathcal{T}_{accept}} \mathrm{rank}(p_\tau q_\tau^T) = |\mathcal{T}_{accept}| \leq 2^c.$$

Therefore, $\mathrm{rank}_\epsilon(M) \leq \mathrm{rank}(P) \leq 2^c$.

Part 3. Let $r = \mathrm{rank}(M)$ and $k = \mathrm{rank}_{1/3}(M)$. We aim to show that the deterministic communication complexity $D(M)$ is bounded by $O(\sqrt{k} \log^2 r)$.

We utilize the structural result by Lovett regarding low-rank matrices.

Lemma D.6 (Lovett's Geometric Lemma)**.** *Let A be a real matrix of rank k. There exists a combinatorial rectangle $R = X' \times Y'$ with measure $\mu(R) \geq 2^{-O(\sqrt{k})}$ such that A restricted to R has constant sign (except possibly on entries where $A_{i,j} = 0$, but here A approximates a Boolean matrix so entries are bounded away from 0).*

Since $k = \mathrm{rank}_{1/3}(M)$, there exists a matrix A with $\mathrm{rank}(A) = k$ such that $\|A - M\|_\infty \leq 1/3$. Since M is Boolean, for any (x, y), if $M_{x,y} = 0$ then $A_{x,y} \in [-1/3, 1/3]$, and if $M_{x,y} = 1$ then $A_{x,y} \in [2/3, 4/3]$. In particular, the sign of $A_{x,y} - 1/2$ determines $M_{x,y}$. Applying Lovett's lemma to the matrix $A - \frac{1}{2}J$ (which has rank at most $k + 1$), there exists a monochromatic rectangle R for M (where M is constant) with size at least $2^{-O(\sqrt{k})}|X \times Y|$.

We now use the standard recursive protocol for the Log-Rank Conjecture context (due to Nisan and Wigderson, and refined by Kotlov). The protocol proceeds as follows: 1. Find a large monochromatic

rectangle R in the current submatrix M'. 2. If the inputs (x, y) belong to R, output the value of M on R. 3. Otherwise, partition the space outside R and recurse.

However, a simple partition does not guarantee small depth. Instead, we use the rank of the matrix as a potential function. A known theorem states that if every submatrix of M contains a monochromatic rectangle of relative size at least $2^{-\alpha}$, then

$$D(M) \leq O(\alpha \log \operatorname{rank}(M)).$$

Note that for any submatrix M' of M, $\operatorname{rank}_{1/3}(M') \leq \operatorname{rank}_{1/3}(M) = k$, because the restriction of the approximating matrix A to the submatrix yields an approximation of rank at most k. Therefore, every submatrix of M contains a monochromatic rectangle of relative size $2^{-O(\sqrt{k})}$. Setting $\alpha = O(\sqrt{k})$, we obtain:

$$D(M) \leq O(\sqrt{k} \log \operatorname{rank}(M)).$$

This bound is $O(\sqrt{\operatorname{rank}_{1/3}(M)} \log \operatorname{rank}(M))$. The statement in the theorem allows for a factor of $\log^2 \operatorname{rank}(M)$. Since $\log r \leq \log^2 r$ for $r \geq 2$, our derived bound is strictly stronger than (and thus implies) the required bound:

$$D(M) \leq O\left(\sqrt{\operatorname{rank}_{1/3}(M)} \log^2 \operatorname{rank}(M)\right).$$

□

E Problems in Chapter 4

E.1 Problem 4.1

E.1.1 Problem Statement

Theorem E.1. *Suppose there are k parties in the number-on-forehead model. The i-th party has the bit $X_i \in \{0, 1\}$ written on their forehead, and $X_1, \ldots, X_k$ are sampled independently and uniformly at random. There exists a protocol for each party to privately write down a guess for the bit on their own forehead, without* any *communication, in such a way that the probability that all parties guess correctly is* $1/2$.

E.1.2 GPT

Proof. Fix in advance the following deterministic protocol. For each $i \in \{1, \ldots, k\}$, upon observing the bits $\{X_j : j \neq i\}$, the i-th party writes down the guess

$$G_i := \bigoplus_{j \neq i} X_j,$$

where $\oplus$ denotes addition modulo 2 (equivalently, XOR). This protocol uses no communication, since G_i depends only on the bits visible to party i.

Let $Y := \bigoplus_{j=1}^{k} X_j$. We claim that the event that all parties guess correctly is exactly the event $\{Y = 0\}$.

First assume that all parties guess correctly. Then for every i we have

$$X_i = G_i = \bigoplus_{j \neq i} X_j.$$

XORing both sides with $\bigoplus_{j \neq i} X_j$ and using associativity and the identity $a \oplus a = 0$, we obtain

$$X_i \oplus \bigoplus_{j \neq i} X_j = 0,$$

that is,

$$\bigoplus_{j=1}^{k} X_j = 0.$$

Hence $\{\forall i,\ G_i = X_i\} \subseteq \{Y = 0\}$.

Conversely, assume that $Y = 0$, i.e., $\bigoplus_{j=1}^{k} X_j = 0$. Fix any i. XORing the equality $\bigoplus_{j=1}^{k} X_j = 0$ with $\bigoplus_{j\neq i} X_j$ on both sides yields

$$\left(\bigoplus_{j=1}^{k} X_j\right) \oplus \left(\bigoplus_{j\neq i} X_j\right) = 0 \oplus \left(\bigoplus_{j\neq i} X_j\right).$$

By cancellation of equal terms under XOR, the left-hand side simplifies to X_i, so we get

$$X_i = \bigoplus_{j\neq i} X_j = G_i.$$

Since i was arbitrary, it follows that $\{Y = 0\} \subseteq \{\forall i,\ G_i = X_i\}$. Therefore,

$$\{\forall i,\ G_i = X_i\} = \{Y = 0\}.$$

It remains to compute $\Pr[Y = 0]$. Let $S := \bigoplus_{j=1}^{k-1} X_j$, which is a $\{0, 1\}$-valued random variable determined by $X_1, \ldots, X_{k-1}$ and hence independent of X_k. Since $Y = S \oplus X_k$, we have

$$\Pr[Y = 0] = \Pr[S \oplus X_k = 0] = \mathbb{E}\big[\Pr[S \oplus X_k = 0 \mid S]\big].$$

For each fixed value $s \in \{0, 1\}$, the event $\{s \oplus X_k = 0\}$ is equivalent to $\{X_k = s\}$, so

$$\Pr[S \oplus X_k = 0 \mid S = s] = \Pr[X_k = s] = \frac{1}{2},$$

because X_k is uniform on $\{0, 1\}$. Hence $\Pr[S \oplus X_k = 0 \mid S] = \frac{1}{2}$ almost surely, and therefore

$$\Pr[Y = 0] = \mathbb{E}\left[\frac{1}{2}\right] = \frac{1}{2}.$$

Combining this with the established identity $\{\forall i,\ G_i = X_i\} = \{Y = 0\}$ gives

$$\Pr[\forall i,\ G_i = X_i] = \frac{1}{2},$$

which proves the theorem. □

E.1.3 Claude

Proof. We construct an explicit protocol and verify that it achieves the desired probability.

Define the protocol as follows: Each party i (for $i = 1, \ldots, k$) guesses that their own bit X_i equals the parity of all the other bits, that is,

$$\widehat{X}_i = X_1 \oplus X_2 \oplus \cdots \oplus X_{i-1} \oplus X_{i+1} \oplus \cdots \oplus X_k = \bigoplus_{j\neq i} X_j$$

where $\oplus$ denotes addition modulo 2.

Note that this protocol requires no communication: party i can see all bits except X_i (since X_i is on their own forehead), and thus can compute $\bigoplus_{j\neq i} X_j$ directly.

We now compute the probability that all parties guess correctly. All parties guess correctly if and only if $\widehat{X}_i = X_i$ for all $i = 1, \ldots, k$, which occurs if and only if

$$X_i = \bigoplus_{j\neq i} X_j \quad \text{for all } i \in \{1, \ldots, k\}.$$

Let $S = X_1 \oplus X_2 \oplus \cdots \oplus X_k$ denote the parity of all bits. For any fixed i, we have

$$\bigoplus_{j \neq i} X_j = S \oplus X_i.$$

Therefore, the condition $X_i = \bigoplus_{j \neq i} X_j$ is equivalent to $X_i = S \oplus X_i$, which simplifies to $X_i \oplus X_i = S$, giving us $0 = S$.

Thus, all parties guess correctly if and only if $S = 0$, that is, if and only if $X_1 \oplus X_2 \oplus \cdots \oplus X_k = 0$.

Since $X_1, \ldots, X_k$ are sampled independently and uniformly at random from $\{0, 1\}$, the sum $S = X_1 \oplus X_2 \oplus \cdots \oplus X_k$ is also uniformly distributed over $\{0, 1\}$. To see this, note that for any fixed values of $X_2, \ldots, X_k$, the value of S depends on X_1 as $S = X_1 \oplus (X_2 \oplus \cdots \oplus X_k)$. Since X_1 is uniform and independent, S takes each value in $\{0, 1\}$ with probability $1/2$.

Therefore,

$$\mathbb{P}[\text{all parties guess correctly}] = \mathbb{P}[S = 0] = \frac{1}{2}.$$

This completes the proof. □

E.1.4 Gemini

Proof. Let k be a positive integer representing the number of parties. Let the bits written on the foreheads be represented by the random variables $X_1, \ldots, X_k$, where each $X_i \in \{0, 1\}$. By the problem statement, these variables are independent and uniformly distributed. Thus, for any realization $(x_1, \ldots, x_k) \in \{0, 1\}^k$, the probability $\Pr[X_1 = x_1, \ldots, X_k = x_k] = 2^{-k}$.

In the number-on-forehead model, party i observes the set of values $\{X_j : j \neq i\}$ but does not observe X_i. We propose the following deterministic protocol: each party i computes the sum modulo 2 of the bits they see and adopts this sum as their guess G_i. Formally, for each $i \in \{1, \ldots, k\}$, let

$$G_i = \sum_{j \neq i} X_j \pmod 2.$$

This protocol requires no communication, as G_i depends exclusively on the information available to party i.

We now determine the condition under which a specific party i guesses correctly. The guess is correct if and only if $G_i = X_i$. Substituting the definition of G_i, this equality holds if and only if:

$$\sum_{j \neq i} X_j \equiv X_i \pmod 2.$$

Adding X_i to both sides of the congruence, and observing that $2X_i \equiv 0 \pmod 2$ for any integer X_i, we obtain the equivalent condition:

$$\sum_{j \neq i} X_j + X_i \equiv 0 \pmod 2 \implies \sum_{j=1}^{k} X_j \equiv 0 \pmod 2.$$

Let $S = \sum_{j=1}^{k} X_j \pmod 2$ denote the parity of the total sum of the bits. The derivation above shows that party i guesses correctly if and only if $S = 0$. Since this condition $S = 0$ does not depend on the index i, it follows that if $S = 0$, then $G_i = X_i$ for all $i \in \{1, \ldots, k\}$. Conversely, if $S = 1$, then $G_i \neq X_i$ for all i.

Therefore, the event that all parties guess correctly is exactly the event that the total sum of the bits is even ($S = 0$). We calculate the probability of this event. Let $S' = \sum_{j=1}^{k-1} X_j \pmod 2$. Then $S = S' + X_k \pmod 2$. Since X_k is uniformly distributed on $\{0, 1\}$ and is independent of S', we have:

$$\Pr[S = 0] = \Pr[S' + X_k \equiv 0 \pmod 2] = \Pr[X_k \equiv -S' \pmod 2] = \Pr[X_k = S'].$$

By the law of total probability conditioning on S':

$$\Pr[X_k = S'] = \Pr[X_k = 0 \mid S' = 0] \Pr[S' = 0] + \Pr[X_k = 1 \mid S' = 1] \Pr[S' = 1].$$

Using the independence of X_k and S', and the fact that $\Pr[X_k = 0] = \Pr[X_k = 1] = 1/2$:

$$\Pr[S = 0] = \frac{1}{2}\Pr[S' = 0] + \frac{1}{2}\Pr[S' = 1] = \frac{1}{2}(\Pr[S' = 0] + \Pr[S' = 1]) = \frac{1}{2}.$$

Thus, the probability that all parties guess correctly using this protocol is $1/2$. □

E.1.5 Grok

Proof. Consider the inputs $X_1, \ldots, X_k \in \{0, 1\}$ as elements of the field $\mathbb{F}_2$, where addition is modulo 2. Define the protocol as follows: the i-th party, upon observing X_j for all $j \neq i$, computes the guess $G_i = \sum_{j \neq i} X_j$, where the sum is taken in $\mathbb{F}_2$.

We claim that all parties guess correctly if and only if $\sum_{j=1}^{k} X_j = 0$ in $\mathbb{F}_2$. To see this, suppose first that $\sum_{j=1}^{k} X_j = 0$. Then, for each i, we have $X_i = \sum_{j=1}^{k} X_j - X_i = -X_i = \sum_{j \neq i} X_j = G_i$, since $-1 = 1$ in $\mathbb{F}_2$. Thus, $G_i = X_i$ for all i.

Conversely, suppose $G_i = X_i$ for all i. In particular, for a fixed i, $X_i = \sum_{j \neq i} X_j$, so $\sum_{j=1}^{k} X_j = \sum_{j \neq i} X_j + X_i = X_i + X_i = 0$ in $\mathbb{F}_2$.

Since $X_1, \ldots, X_k$ are independent and uniform over $\{0, 1\}$, the sum $\sum_{j=1}^{k} X_j$ in $\mathbb{F}_2$ is 0 with probability exactly $\frac{1}{2}$ (for $k \geq 1$, as the distribution is balanced). Therefore, the probability that all parties guess correctly is $\frac{1}{2}$. □

E.2 Problem 4.2

E.2.1 Problem Statement

Theorem E.2. *Define the generalized inner product function* GIP *as follows. For k inputs $x_1, \ldots, x_k \in \{0, 1\}^n$,*

$$\mathsf{GIP}(x) = \sum_{j=1}^{n} \prod_{i=1}^{k} x_{i,j} \bmod 2.$$

This exercise outlines a number-on-forehead GIP *protocol using $O(n/2^k + k)$ bits. It is convenient to think about the input X as a $k \times n$ matrix with rows corresponding to $x_1, \ldots, x_k$.*

- *For $\ell \in \{0, 1, \ldots, k-1\}$ define c_ℓ as the number of columns in X that start with ℓ zeroes, followed by either a one or zero, followed by $k - \ell - 1$ ones. Prove that $\sum_{\ell=0}^{k-1} c_\ell = \mathsf{GIP}(x) \bmod 2$.*
- *Suppose the parties know a string $z \in \{0, 1\}^k$ with the property that no column of X is equal to z. Find a protocol to compute $\mathsf{GIP}(x)$ using $O(k)$ bits* assuming the parties know z.
- *Exhibit an overall protocol for* GIP *by showing that the parties can agree upon a vector z and communicate to determine $c_\ell \pmod 2$ using $O(n/2^k + k)$ bits.*

E.2.2 GPT

Proof. We are given the generalized inner product function GIP defined as follows for k inputs $x_1, \ldots, x_k \in \{0, 1\}^n$:

$$\mathsf{GIP}(x) = \sum_{j=1}^{n} \prod_{i=1}^{k} x_{i,j} \mod 2.$$

We are also provided with the definition of c_ℓ, the number of columns in the matrix X that start with ℓ zeroes, followed by a one or zero, and then followed by $k - \ell - 1$ ones. We need to prove that:

$$\sum_{\ell=0}^{k-1} c_\ell = \mathsf{GIP}(x) \mod 2.$$

Step 1: Understanding the structure of c_ℓ Let X be the $k \times n$ matrix where each row corresponds to one of the inputs $x_1, \ldots, x_k$, and each column corresponds to an individual bit of these inputs. Denote the columns of X as $X_j = (x_{1,j}, x_{2,j}, \ldots, x_{k,j})$ for $j \in \{1, \ldots, n\}$. The value of c_ℓ counts the number

of columns where the first ℓ entries are zeroes, the next entry is either a zero or one, and the remaining $k-\ell-1$ entries are ones.

Step 2: Expressing c_ℓ more explicitly Let X_j be a column of X. For each $\ell \in \{0, 1, \ldots, k-1\}$, define the following condition:

- The first ℓ entries of X_j are zeroes, i.e., $x_{1,j} = x_{2,j} = \cdots = x_{\ell,j} = 0$. - The next entry, $x_{\ell+1,j}$, can be either zero or one. - The remaining $k-\ell-1$ entries, i.e., $x_{\ell+2,j}, x_{\ell+3,j}, \ldots, x_{k,j}$, are all ones.

Thus, a column X_j contributes to c_ℓ if it matches this structure. Specifically, for each column X_j, the condition is that $x_{1,j} = x_{2,j} = \cdots = x_{\ell,j} = 0$, $x_{\ell+1,j}$ can be either 0 or 1, and $x_{\ell+2,j}, x_{\ell+3,j}, \ldots, x_{k,j}$ are all 1.

Step 3: Relating c_ℓ to the generalized inner product We aim to show that the sum of the c_ℓ's modulo 2 is equal to $\mathsf{GIP}(x)$ modulo 2. The expression for $\mathsf{GIP}(x)$ is:

$$\mathsf{GIP}(x) = \sum_{j=1}^{n} \prod_{i=1}^{k} x_{i,j} \mod 2.$$

The product $\prod_{i=1}^{k} x_{i,j}$ is 1 if and only if all entries $x_{1,j}, x_{2,j}, \ldots, x_{k,j}$ are 1, and 0 otherwise. Therefore, $\mathsf{GIP}(x)$ counts the number of columns in X where all entries are 1, and we are interested in this sum modulo 2.

The structure of c_ℓ can be viewed as counting certain patterns in the columns, specifically those with the first ℓ entries as zeroes, the next as either zero or one, and the remaining as ones. Thus, the sum of c_ℓ's counts the number of columns that meet the conditions defined by the c_ℓ's, and modulo 2, this sum corresponds to the sum of the products $\prod_{i=1}^{k} x_{i,j}$ for all j. Therefore, we have:

$$\sum_{\ell=0}^{k-1} c_\ell = \mathsf{GIP}(x) \mod 2.$$

This completes the proof of the first part of the theorem.

Step 4: Finding a protocol to compute $\mathsf{GIP}(x)$ assuming the parties know z Suppose the parties know a string $z \in \{0,1\}^k$ such that no column of X is equal to z. In this case, we wish to compute $\mathsf{GIP}(x)$ using $O(k)$ bits.

Given the knowledge of z, the parties can use the fact that z is not equal to any column of X to determine the value of $\mathsf{GIP}(x)$. Specifically, they can use z to identify which columns contribute to the sum of the products $\prod_{i=1}^{k} x_{i,j}$. Since no column of X is equal to z, the protocol can exploit the structure of the inputs and the value of z to efficiently compute $\mathsf{GIP}(x)$ using $O(k)$ bits.

Step 5: Overall protocol for GIP Finally, to compute $\mathsf{GIP}(x)$, the parties can follow the following protocol:

1. Agree upon a vector $z \in \{0,1\}^k$ such that no column of X is equal to z. 2. Use the knowledge of z and communicate to determine $c_\ell \pmod 2$ for all $\ell \in \{0, 1, \ldots, k-1\}$. 3. The sum of $c_\ell \pmod 2$ for all ℓ gives $\mathsf{GIP}(x) \pmod 2$.

This protocol requires $O(n/2^k + k)$ bits, as it involves determining the values of $c_\ell \pmod 2$ and using the value of z to efficiently compute the desired result.

Thus, the overall protocol can compute $\mathsf{GIP}(x)$ using $O(n/2^k + k)$ bits, as required.

□

E.2.3 Claude

Proof. We prove each part of the theorem in turn.

Part 1: We prove that $\sum_{\ell=0}^{k-1} c_\ell \equiv \mathsf{GIP}(x) \pmod 2$.

For each column $j \in \{1, \ldots, n\}$, let ℓ_j denote the number of leading zeros in column j before the first one appears (if any). More precisely, ℓ_j is the smallest index i such that $x_{i,j} = 1$, or $\ell_j = k$ if the column is all zeros.

Observe that column j contributes to $\mathsf{GIP}(x)$ if and only if $\prod_{i=1}^{k} x_{i,j} = 1$, which occurs precisely when all entries in column j are ones. This happens when $\ell_j = 0$ and all remaining entries are ones.

Now consider the contribution of column j to c_ℓ for various ℓ. Column j is counted in c_ℓ if it has exactly ℓ leading zeros, followed by a one (at position $\ell + 1$), and then $k - \ell - 1$ ones in the remaining positions. This means column j contributes to exactly one c_ℓ value, namely c_{ℓ_j}, if and only if $\ell_j < k$ and all entries after position $\ell_j + 1$ are ones.

For a column with all ones, we have $\ell_j = 0$ and it contributes to c_0. For a column that does not have all ones but has the pattern of ℓ zeros followed by a one and then all ones, it contributes to c_ℓ for $\ell > 0$.

The key observation is that each column contributes to at most one c_ℓ, and it contributes to some c_ℓ with $\ell < k$ if and only if it has the form: ℓ zeros, then a one, then $k - \ell - 1$ ones. The product $\prod_{i=1}^{k} x_{i,j} = 1$ if and only if $\ell_j = 0$ and all entries are ones, which means the column contributes to c_0.

Therefore, $\sum_{\ell=0}^{k-1} c_\ell$ counts, modulo 2, exactly those columns where the product is 1, which equals $\mathsf{GIP}(x) \pmod 2$.

Part 2: Given a string $z \in \{0,1\}^k$ such that no column of X equals z, we construct a protocol using $O(k)$ bits.

Since no column equals z, for each column j, there exists at least one position i where $x_{i,j} \neq z_i$. Player i (who sees all rows except row i) can determine for each column j whether that column could potentially equal z given the visible rows.

For each $\ell \in \{0, \ldots, k-1\}$, player $\ell + 1$ can see all rows except row $\ell + 1$. This player can compute c'_ℓ, the number of columns that have ℓ zeros in the first ℓ positions, then potentially any value in position $\ell + 1$, and then all ones in positions $\ell + 2$ through k. Since player $\ell + 1$ knows that no column equals z, they can determine $c_\ell \pmod 2$ by checking which columns match the pattern.

Each player $\ell + 1$ broadcasts one bit: $c_\ell \pmod 2$. This requires k bits total. The parties then compute $\sum_{\ell=0}^{k-1} c_\ell \pmod 2 = \mathsf{GIP}(x) \pmod 2$.

Part 3: We exhibit the overall protocol using $O(n/2^k + k)$ bits.

First, the parties agree on a vector $z \in \{0,1\}^k$ such that no column of X equals z. Since there are 2^k possible vectors and n columns, by the pigeonhole principle, if $n < 2^k$, there exists such a z.

The parties use the following protocol: Each player i proposes a candidate z by finding a vector that does not appear among the columns visible to player i. Since player i sees $k - 1$ rows, they see at most n distinct column patterns. If $n < 2^k$, player i can find such a z.

To agree on z, the parties can use a deterministic rule: they each compute their candidate and select the lexicographically smallest z that is valid for all players. To verify this, each player broadcasts $O(\log 2^k) = O(k)$ bits to describe their candidate z, requiring $O(k^2)$ bits total, which is $O(k)$ when k is constant.

Alternatively, the parties can use a more efficient approach: they jointly compute the set of all columns and find a missing vector. This requires communicating information about $O(n/2^k)$ columns on average.

Once z is agreed upon using $O(n/2^k + k)$ bits, the parties apply the protocol from Part 2, using an additional $O(k)$ bits to compute $\mathsf{GIP}(x)$.

The total communication is $O(n/2^k + k)$ bits. □

E.2.4 Gemini

Proof. We address the three parts of the theorem sequentially.

Part 1: The identity for $\sum c_\ell$. Let $N(v)$ denote the number of columns in X equal to the vector $v \in \{0,1\}^k$. Recall the definition of c_ℓ for $\ell \in \{0, \ldots, k-1\}$. A column $x_{\cdot,j}$ contributes to c_ℓ if and only if:

$$x_{1,j} = \cdots = x_{\ell,j} = 0, \quad x_{\ell+2,j} = \cdots = x_{k,j} = 1,$$

and $x_{\ell+1,j} \in \{0,1\}$. Let $u_\ell \in \{0,1\}^k$ be the vector consisting of ℓ zeros followed by $k - \ell$ ones (i.e., $u_\ell = 0^\ell 1^{k-\ell}$). The condition for c_ℓ specifies that the prefix of length ℓ matches u_ℓ and the suffix of length $k - \ell - 1$ matches u_ℓ. The bit at index $\ell + 1$ is unconstrained. Thus, the set of columns counted by c_ℓ is exactly the union of the set of columns equal to u_ℓ (where $x_{\ell+1,j} = 1$) and the set of columns equal to $u_{\ell+1}$ (where $x_{\ell+1,j} = 0$, noting that $u_{\ell+1} = 0^{\ell+1}1^{k-\ell-1}$). Therefore, modulo 2, we have:

$$c_\ell \equiv N(u_\ell) + N(u_{\ell+1}) \pmod 2.$$

Summing this over $\ell = 0$ to $k-1$:

$$\sum_{\ell=0}^{k-1} c_\ell \equiv \sum_{\ell=0}^{k-1} (N(u_\ell) + N(u_{\ell+1})) \pmod 2.$$

This is a telescoping sum. The terms $N(u_1), \ldots, N(u_{k-1})$ appear twice and cancel out modulo 2. The remaining terms are $N(u_0)$ and $N(u_k)$. By definition, $u_0 = 1^k$ (the all-one vector) and $u_k = 0^k$ (the all-zero vector). Thus,

$$\sum_{\ell=0}^{k-1} c_\ell \equiv N(1^k) + N(0^k) \pmod 2.$$

By definition, $\mathsf{GIP}(x) = N(1^k) \pmod 2$. The statement $\sum_{\ell=0}^{k-1} c_\ell = \mathsf{GIP}(x) \pmod 2$ holds if and only if $N(0^k) \equiv 0 \pmod 2$. In the context of the subsequent parts where we can transform the input or assume specific properties of z, this identity establishes the link between the "telescoping counts" and the target GIP value.

Part 2: Protocol knowing z. Suppose the parties know a vector $z \in \{0,1\}^k$ such that $N(z) = 0$. We wish to compute $\mathsf{GIP}(x) \equiv N(1^k) \pmod 2$. Let $v^{(0)}, v^{(1)}, \ldots, v^{(m)}$ be a sequence of vectors in $\{0,1\}^k$ such that $v^{(0)} = 1^k$, $v^{(m)} = z$, and for each $t \in \{0, \ldots, m-1\}$, $v^{(t)}$ and $v^{(t+1)}$ differ in exactly one coordinate. We can choose such a path with $m \leq k$ (the Hamming distance). Let p_{t+1} be the index where $v^{(t)}$ and $v^{(t+1)}$ differ. For each step t, define C_t as the number of columns in X that match the common bits of $v^{(t)}$ and $v^{(t+1)}$ on all indices except p_{t+1}. Specifically, C_t counts columns y such that $y_i = v_i^{(t)}$ for all $i \neq p_{t+1}$. Similar to Part 1, $C_t = N(v^{(t)}) + N(v^{(t+1)})$. Player p_{t+1} can compute $C_t \pmod 2$ locally, because she sees all rows except row p_{t+1}, and the condition for C_t depends only on rows $i \neq p_{t+1}$. The parties execute the following protocol: 1. The parties agree on the path $v^{(0)}, \ldots, v^{(m)}$ from 1^k to z. 2. For each $t \in \{0, \ldots, m-1\}$, Player p_{t+1} computes $b_t = C_t \pmod 2$ and broadcasts it. 3. The parties compute $S = \sum_{t=0}^{m-1} b_t \pmod 2$. Observe that:

$$S \equiv \sum_{t=0}^{m-1} (N(v^{(t)}) + N(v^{(t+1)})) \equiv N(v^{(0)}) + N(v^{(m)}) \equiv N(1^k) + N(z) \pmod 2.$$

Since $N(z) = 0$, we have $S \equiv N(1^k) \equiv \mathsf{GIP}(x) \pmod 2$. The total communication is $m \leq k$ bits, which is $O(k)$.

Part 3: Overall protocol. To handle the general case, the parties partition the n columns into blocks where they can ensure a missing vector exists. Protocol: 1. The parties partition the set of columns $\{1, \ldots, n\}$ into $B = \lceil n/(2^k - 1) \rceil$ blocks $J_1, \ldots, J_B$, each of size at most $2^k - 1$. 2. For each block $r \in \{1, \ldots, B\}$, let $X^{(r)}$ be the submatrix of X restricted to columns in J_r. Since $|J_r| < 2^k$, there exists at least one vector $z^{(r)} \in \{0,1\}^k$ that does not appear as a column in $X^{(r)}$. 3. The parties agree on such a vector $z^{(r)}$ for each block. (This can be done using shared randomness to select candidates and communicating to verify absence, contributing to the $O(n/2^k)$ term, or deterministically if allowed by the model specifics). 4. For each block r, the parties define a path from 1^k to $z^{(r)}$ and determine the required bits $C_t^{(r)}$ as in Part 2. 5. Instead of broadcasting bits for every block separately, the parties aggregate the counts. Each Player i computes the sum modulo 2 of all $C_t^{(r)}$ terms for which she is responsible (i.e., where the path edge flips bit i) across all blocks r. 6. Each Player i broadcasts her aggregate bit. 7. The sum of these k bits equals $\sum_{r=1}^{B} \mathsf{GIP}(X^{(r)}) \pmod 2$, which is $\mathsf{GIP}(X)$.

Complexity Analysis: The aggregation step (Steps 5-6) requires k bits of communication. The cost to agree on the vectors $z^{(r)}$ (Step 3) depends on the density. With shared randomness, the parties can select a random z. The expected number of occurrences of a random z in a block of size M is $M/2^k$. If the parties simply communicate the locations of these occurrences to "correct" the count (effectively treating z as the target for a correction term), the cost is proportional to the number of blocks times the expected occurrences, or directly related to $n/2^k$. Specifically, standard protocols for finding a zero-count vector or correcting for a low-count vector in this setting operate within $O(n/2^k)$ bits. Thus, the total communication is $O(n/2^k + k)$. □

E.3 Problem 4.3

E.3.1 Problem Statement

Theorem E.3. *Let g be a function in the number-on-forehead model. Assume that the communication complexity of g is strictly greater than the number of parties. Define c_n to be the communication required to compute the AND of n copies of g. Then*

$$\lim_{n\to\infty} c_n = \infty.$$

E.3.2 GPT

Proof. Let g be a function in the number-on-forehead model. Denote the communication complexity of g by $C(g)$, which represents the number of bits exchanged between the parties to compute g. We are given that $C(g) > m$, where m is the number of parties involved in the number-on-forehead model.

Now, consider the function g^n, which is the AND of n copies of g, i.e., the function that outputs 1 if and only if all n copies of g output 1. The communication complexity of g^n, denoted by c_n, is the communication required to compute the AND of these n copies.

We aim to show that $\lim_{n\to\infty} c_n = \infty$. To do so, we will show that for large n, the communication complexity c_n grows without bound.

Step 1: Lower Bound on c_n We first note that the communication complexity of computing g^n is at least the communication complexity of computing g for each of the n copies independently. Since the AND operation requires that all copies of g output 1, the communication complexity of computing g^n cannot be smaller than the communication complexity of computing each g individually.

Thus, we have the inequality

$$c_n \geq n \cdot C(g).$$

Step 2: Behavior of c_n as $n \to \infty$ Given that $C(g) > m$, it follows that

$$c_n \geq n \cdot C(g) > n \cdot m.$$

As $n \to \infty$, this lower bound grows without bound. Specifically, for any large number N, there exists an $n \geq N$ such that

$$c_n > N.$$

Thus, as n increases, c_n becomes arbitrarily large.

Step 3: Conclusion Since $c_n \geq n \cdot C(g)$ and $C(g) > m$, it follows that $c_n \to \infty$ as $n \to \infty$. Therefore, we conclude that

$$\lim_{n\to\infty} c_n = \infty.$$

□

E.3.3 Claude

Proof. Let k denote the number of parties in the number-on-forehead model, and let $\mathrm{CC}(g)$ denote the communication complexity of g. By hypothesis, $\mathrm{CC}(g) > k$.

We proceed by contradiction. Suppose that $\lim_{n\to\infty} c_n \neq \infty$. Then there exists a constant $C > 0$ such that $c_n \leq C$ for infinitely many values of n. By passing to a subsequence if necessary, we may assume without loss of generality that $c_n \leq C$ for all $n \geq 1$.

Consider the function $g^{(n)}(x_1, \ldots, x_n) = g(x_1) \wedge g(x_2) \wedge \cdots \wedge g(x_n)$, which computes the AND of n copies of g. By definition, $c_n = \mathrm{CC}(g^{(n)})$.

We now establish a key observation. In the number-on-forehead model, if a protocol computes $g^{(n)}$ with communication c_n, then for any fixed values of $n-1$ of the inputs, say $x_2, \ldots, x_n$, the protocol induces a protocol for computing $g(x_1)$ conditioned on these fixed values.

More precisely, fix any inputs $x_2, \ldots, x_n$ such that $g(x_i) = 1$ for all $i \in \{2, \ldots, n\}$. Then $g^{(n)}(x_1, \ldots, x_n) = g(x_1)$. The protocol for $g^{(n)}$ restricted to these fixed inputs yields a protocol for computing $g(x_1)$ with communication at most $c_n \leq C$.

However, this contradicts the assumption that $\mathrm{CC}(g) > k$. To see this more carefully, we note that the protocol for $g^{(n)}$ uses at most C bits of communication. When we fix $x_2, \ldots, x_n$, the resulting protocol for $g(x_1)$ uses at most C bits of communication. Since this holds for all sufficiently large n, and the communication complexity of g is an intrinsic property of the function independent of n, we must have $\mathrm{CC}(g) \leq C$.

But we can choose C to be any upper bound on the subsequence $\{c_n\}$. If $\limsup_{n\to\infty} c_n < \infty$, then there exists such a finite C, leading to $\mathrm{CC}(g) \leq C < \infty$.

To complete the contradiction, we use a direct sum principle for the number-on-forehead model. By results in communication complexity theory, if $\mathrm{CC}(g) > k$, then for the AND function applied to n independent copies of g, we have $c_n \geq \Omega(\log n)$ or stronger lower bounds depending on the specific properties of g. More generally, the communication complexity of computing the AND of n copies cannot remain bounded as $n \to \infty$ when the base function has communication complexity exceeding the number of parties.

Therefore, our assumption that $\lim_{n\to\infty} c_n \neq \infty$ leads to a contradiction. We conclude that $\lim_{n\to\infty} c_n = \infty$. □

E.3.4 Gemini

Proof. Let k denote the number of parties. Let $X_1, \ldots, X_k$ be the input spaces for the k parties. A function $g : X_1 \times \cdots \times X_k \to \{0, 1\}$ is computed in the number-on-forehead (NOF) model. The communication complexity of g, denoted $D(g)$, is the minimum number of bits exchanged in a deterministic protocol that computes g. We are given that $D(g) > k$.

Let g^n denote the function that computes the AND of n copies of g. The input to g^n consists of n tuples $(x^{(1)}, \ldots, x^{(n)})$, where each $x^{(j)} \in X_1 \times \cdots \times X_k$. The value of the function is

$$g^n(x^{(1)}, \ldots, x^{(n)}) = \bigwedge_{j=1}^{n} g(x^{(j)}).$$

Let $c_n = D(g^n)$. We wish to show that $\lim_{n\to\infty} c_n = \infty$.

We utilize the concept of nondeterministic communication complexity. Let $N^1(f)$ denote the nondeterministic communication complexity of a function f with respect to the output value 1. It is a standard result in communication complexity that for any function f,

$$D(f) \geq N^1(f).$$

Therefore, $c_n \geq N^1(g^n)$.

The value $N^1(f)$ is related to the covering number of the set $f^{-1}(1)$. A subset $R \subseteq X_1 \times \cdots \times X_k$ is called a *cylinder intersection* if it can be expressed as $R = \bigcap_{i=1}^{k} A_i$, where each set A_i has the property that membership does not depend on the i-th coordinate (i.e., if two inputs differ only in the i-th coordinate, they are either both in A_i or both not in A_i). In the NOF model, monochromatic rectangles for the value 1 correspond to cylinder intersections.

Let $\mathrm{cov}(f)$ be the minimum number of cylinder intersections required to cover the set $f^{-1}(1)$. Then,

$$N^1(f) = \lceil \log_2 \mathrm{cov}(f) \rceil.$$

Consider the function g^n. The set $(g^n)^{-1}(1)$ is the Cartesian product of n copies of $g^{-1}(1)$ (viewed in the appropriate product space). A fundamental property of the covering number for product sets is that it is multiplicative. Specifically,

$$\mathrm{cov}(g^n) = (\mathrm{cov}(g))^n.$$

Taking logarithms, we have:

$$N^1(g^n) \geq \log_2 \mathrm{cov}(g^n) = n \log_2 \mathrm{cov}(g).$$

Combining this with the deterministic bound, we obtain:

$$c_n \geq n \log_2 \mathrm{cov}(g).$$

We now analyze $\text{cov}(g)$. Since $\text{cov}(g)$ is a positive integer, there are two cases:

Case 1: $\text{cov}(g) = 1$. If $\text{cov}(g) = 1$, then $g^{-1}(1)$ is a single cylinder intersection. Let $g^{-1}(1) = \bigcap_{i=1}^{k} A_i$, where A_i does not depend on the i-th coordinate. We can construct a deterministic protocol for g as follows: Each party P_i knows all input coordinates except the i-th. Since membership in A_i does not depend on the i-th coordinate, P_i can determine whether the input x belongs to A_i. Protocol: 1. For each $i \in \{1, \ldots, k\}$, party P_i broadcasts a single bit b_i, where $b_i = 1$ if $x \in A_i$ and $b_i = 0$ otherwise. 2. All parties compute the output as $\bigwedge_{i=1}^{k} b_i$.

This protocol correctly computes g because $g(x) = 1$ if and only if $x \in \bigcap_{i=1}^{k} A_i$, which is equivalent to $x \in A_i$ for all i. The total communication cost is exactly k bits. Thus, if $\text{cov}(g) = 1$, then $D(g) \leq k$. However, the theorem statement assumes $D(g) > k$. This is a contradiction. Therefore, $\text{cov}(g) \neq 1$.

Case 2: $\text{cov}(g) \geq 2$. Since $\text{cov}(g)$ is an integer, we must have $\text{cov}(g) \geq 2$. Consequently, $\log_2 \text{cov}(g) \geq 1$. Substituting this into our lower bound for c_n:

$$c_n \geq n \log_2 \text{cov}(g) \geq n.$$

Taking the limit as $n \to \infty$:

$$\lim_{n\to\infty} c_n \geq \lim_{n\to\infty} n = \infty.$$

Thus, $\lim_{n\to\infty} c_n = \infty$. □

E.4 Problem 4.4

E.4.1 Problem Statement

Theorem E.4. *A three-party number-on-forehead puzzle demonstrates that unexpected efficiency is sometimes possible.*

Alice has a number $i \in [n]$ on her forehead, Bob has a number $j \in [n]$ on his forehead, and Charlie has a string $x \in \{0,1\}^n$ on his forehead. On input (i, j, x) the goal is for Charlie to output the bit x_k where $k = i + j \bmod n$.

Find a deterministic protocol in which Bob sends one bit to Charlie, and Alice sends $\lceil \frac{n}{2} \rceil$ bits to Charlie. Alice and Bob must send Charlie their message simultaneously. *Charlie is then able to output the correct answer.*

E.4.2 GPT

Proof. Let n be a positive integer, and suppose that Alice, Bob, and Charlie are participating in the number-on-forehead puzzle. Alice has a number $i \in [n]$, Bob has a number $j \in [n]$, and Charlie has a string $x \in \{0,1\}^n$. The goal is for Charlie to output the bit x_k where $k = (i + j) \bmod n$.

We aim to construct a deterministic protocol in which Bob sends one bit to Charlie, and Alice sends $\lceil \frac{n}{2} \rceil$ bits to Charlie. Furthermore, Alice and Bob must send their messages to Charlie simultaneously, and Charlie must be able to correctly compute x_k.

Step 1: Alice's message to Charlie

Alice knows her number i, and she needs to convey enough information to Charlie so that he can compute the value of x_k for $k = (i + j) \bmod n$. Since k depends on both i and j, Alice must send some information about i that allows Charlie to determine k without knowing j.

Alice sends a string of $\lceil \frac{n}{2} \rceil$ bits to Charlie. This string will represent a part of x based on i. Specifically, Alice will send the bits $x_1, x_2, \ldots, x_{\lceil \frac{n}{2} \rceil}$ corresponding to indices in the string x, based on the value of i. Thus, Alice's message consists of the first $\lceil \frac{n}{2} \rceil$ bits of x, and this string will allow Charlie to deduce the value of x_k when combined with Bob's message.

Step 2: Bob's message to Charlie

Bob knows his number j, and he must send one bit to Charlie. Since $k = (i + j) \bmod n$, the value of k depends on both i and j. Therefore, Bob's message should convey sufficient information for Charlie to determine the value of x_k given Alice's message.

Bob sends a single bit, which encodes information about $x_{(i+j) \bmod n}$. This bit allows Charlie to compute the value of x_k by combining it with Alice's message.

Step 3: Charlie's computation

Upon receiving Alice's $\lceil \frac{n}{2} \rceil$ bits and Bob's one bit, Charlie can compute $k = (i + j) \bmod n$. Using this value of k, Charlie looks up x_k from the string x and outputs it as the result.

The key observation is that, while Bob and Alice have very limited communication (Bob sends only one bit and Alice sends $\lceil \frac{n}{2} \rceil$ bits), the combination of their messages allows Charlie to deduce x_k deterministically. This efficient communication scheme exploits the structure of the number-on-forehead puzzle, demonstrating that unexpected efficiency is sometimes possible.

Conclusion

We have constructed a deterministic protocol in which Bob sends one bit to Charlie, and Alice sends $\lceil \frac{n}{2} \rceil$ bits to Charlie. Alice and Bob send their messages simultaneously, and Charlie is able to compute the correct bit x_k for $k = (i + j) \bmod n$. This completes the proof of the theorem. □

E.4.3 Claude

Proof. We construct a deterministic protocol satisfying the requirements.

Let n be given and let $m = \lceil \frac{n}{2} \rceil$. We partition the index set $[n] = \{0, 1, \ldots, n-1\}$ into two disjoint sets S_0 and S_1 where $S_0 = \{0, 1, \ldots, m-1\}$ and $S_1 = \{m, m+1, \ldots, n-1\}$. Note that $|S_0| = m$ and $|S_1| = n - m = \lfloor \frac{n}{2} \rfloor$.

The protocol proceeds as follows:

Bob's message: Bob sees j on Alice's forehead and x on Charlie's forehead. Bob computes the parity bit $b = \bigoplus_{k \in S_0} x_{(k-j) \bmod n}$, where $\bigoplus$ denotes the XOR operation. Bob sends the single bit b to Charlie.

Alice's message: Alice sees j on Bob's forehead and x on Charlie's forehead. For each $k \in S_0$, Alice computes $x_{(k-j) \bmod n}$ and sends these m bits to Charlie. Specifically, Alice sends the string $(x_{(0-j) \bmod n}, x_{(1-j) \bmod n}, \ldots, x_{(m-1-j) \bmod n})$.

Charlie's computation: Charlie receives bit b from Bob and the m-bit string from Alice. Charlie sees i on Alice's forehead and j on Bob's forehead. Charlie computes $k = (i + j) \bmod n$.

We now show that Charlie can determine x_k:

Case 1: If $i \in S_0$, then $k - j = (i + j) - j = i \pmod n$, so $k - j \equiv i \pmod n$ where $i \in S_0$. Since Alice sent $x_{(i-j) \bmod n}$ as part of her message (specifically, as the $(i + 1)$-th bit of her message), Charlie can directly read x_k from Alice's message.

Case 2: If $i \in S_1$, then Charlie needs to recover x_k using Bob's parity bit. Charlie knows all values $x_{(k'-j) \bmod n}$ for $k' \in S_0$ from Alice's message. Bob's bit b equals $\bigoplus_{k' \in S_0} x_{(k'-j) \bmod n}$.

We claim that the sets $\{(k' - j) \bmod n : k' \in S_0\}$ and $\{(k - j) \bmod n\} \cup \{(k' - j) \bmod n : k' \in S_0 \setminus \{k \bmod n\}\}$ partition appropriately. Since $i \in S_1$ and $k = (i+j) \bmod n$, we have $(k-j) \bmod n = i \in S_1$.

Charlie computes $c = \bigoplus_{k' \in S_0} x_{(k'-j) \bmod n}$ using Alice's message. Since $b = c$ by Bob's construction, and since $(k-j) \bmod n \notin \{(k'-j) \bmod n : k' \in S_0\}$ (as $i \in S_1$ and $S_0 \cap S_1 = \emptyset$), Charlie determines x_k by observing that if we define the partition $[n] = \{(k'-j) \bmod n : k' \in S_0\} \cup \{(k'-j) \bmod n : k' \in S_1\}$, then Bob's parity bit encodes information about S_0 shifted by $-j$.

More precisely, Charlie computes $x_k = b \oplus c'$ where $c' = \bigoplus_{k' \in S_0, k' \neq (k-j) \bmod n} x_{(k'-j) \bmod n}$. However, since $(k - j) \bmod n = i \notin S_0$, we have $c' = c$, so this approach requires modification.

Instead, Charlie uses the fact that Bob computed b knowing the target index relative to his position. Since Charlie knows i, j, and all $x_{(k'-j) \bmod n}$ for $k' \in S_0$, and since $i \in S_1$ implies $k = (i + j) \bmod n$ corresponds to an index not covered by Alice's message, Charlie can determine x_k as the unique bit making Bob's parity consistent.

Therefore, in both cases, Charlie correctly outputs x_k. The protocol uses 1 bit from Bob and $\lceil \frac{n}{2} \rceil$ bits from Alice, sent simultaneously. □

E.5 Problem 4.5

E.5.1 Problem Statement

Theorem E.5. *Suppose Alice has a number $i \in [n]$ on her forehead, Bob has a permutation $\pi : [n] \to [n]$ on his forehead, and Charlie has a string $x \in \{0, 1\}^n$ on his forehead. There exists a protocol to compute $x_{\pi(i)}$ where Alice sends Charlie at most $O(n/\log n)$ bits, and Bob sends Charlie at most $n/2$ bits.*

Lemma E.6. *There exist sets $S_1, \ldots, S_n \subseteq [n]$ with the property that $|S_i| \leq n/2$ for all i, and for any permutation $\pi : [n] \to [n]$, one can color $[n]$ with $O(n/\log n)$ colors so that if i, j are assigned the same color, then $\pi(j) \in S_i$.*

E.5.2 GPT

Proof. We will prove the theorem by first proving the lemma, and then using it to construct the desired protocol.

Proof of Lemma:

We are given a permutation $\pi : [n] \to [n]$ and need to construct sets $S_1, S_2, \ldots, S_n \subseteq [n]$ such that for each $i \in [n]$, $|S_i| \leq n/2$ and for any permutation π, there exists a coloring of $[n]$ with $O(n/\log n)$ colors such that if i and j are assigned the same color, then $\pi(j) \in S_i$.

To construct the sets $S_1, S_2, \ldots, S_n$, we proceed as follows:

1. Partition the set $[n]$ into two disjoint subsets, A and B, where $A = \{1, 2, \ldots, \lfloor n/2 \rfloor\}$ and $B = \{\lfloor n/2 \rfloor + 1, \ldots, n\}$. 2. For each $i \in [n]$, define the set $S_i \subseteq [n]$ as follows:

$$S_i = \{j \in A : \pi(j) = i\}.$$

Since π is a permutation, $|S_i| \leq n/2$ for all i by construction.

Now, we need to color $[n]$ such that if i and j are assigned the same color, then $\pi(j) \in S_i$. This is accomplished by the following coloring strategy:

3. Assign each element of A a distinct color. Since $|A| \leq n/2$, there are at most $n/2$ colors needed for this assignment. 4. For each element $j \in B$, assign the same color as the element $i \in A$ such that $\pi(j) \in S_i$. This ensures that if i and j are assigned the same color, then $\pi(j) \in S_i$.

Thus, we have constructed a coloring of $[n]$ with at most $O(n/\log n)$ colors, as required.

Proof of Theorem:

Now, we use the lemma to construct the protocol for computing $x_{\pi(i)}$. Alice has the number $i \in [n]$, Bob has the permutation π, and Charlie has the string $x \in \{0, 1\}^n$.

1. Alice sends Charlie a message containing the color assigned to i according to the coloring constructed in the lemma. This message consists of at most $O(n/\log n)$ bits since there are $O(n/\log n)$ possible colors. 2. Bob sends Charlie the index of i within the set S_i corresponding to the color that Alice sent. This message consists of at most $n/2$ bits, since $|S_i| \leq n/2$.

Given these two messages, Charlie can compute $x_{\pi(i)}$ by using the information from the coloring and the index provided by Bob. Therefore, the protocol works and requires at most $O(n/\log n)$ bits from Alice and $n/2$ bits from Bob, as desired.

Thus, we have proved the theorem. □

E.5.3 Claude

Proof. We prove the theorem by first establishing the lemma, then using it to construct the protocol.

Proof of Lemma:

We construct the sets $S_1, \ldots, S_n$ using a probabilistic argument. For each $i \in [n]$, let S_i be a uniformly random subset of $[n]$ of size exactly $\lfloor n/2 \rfloor$. We will show that with positive probability, these sets satisfy the required property.

For a fixed permutation $\pi : [n] \to [n]$, consider a greedy coloring procedure. We color elements of $[n]$ one by one, assigning to each element i the smallest color c such that for all previously colored elements j with color c, we have $\pi(j) \in S_i$.

Let C_k denote the set of elements assigned color k. For any $i \in C_k$ and $j \in C_k$ with j colored before i, we must have $\pi(j) \in S_i$ by construction. Since $|S_i| \geq n/2$, when we attempt to color element i, color k is unavailable only if there exists some j already assigned color k with $\pi(j) \notin S_i$.

For a fixed element i and color class C_k of size m, the probability that i cannot use color k is at most the probability that $\pi(C_k) \not\subseteq S_i$, which equals $1 - \frac{\binom{n/2}{m}}{\binom{n}{m}}$ when $m \leq n/2$.

By a counting argument, if we use $t = c \cdot n/\log n$ colors for sufficiently large constant c, then each color class has expected size $O(\log n)$. By concentration inequalities, with high probability all color classes have size $O(\log n)$.

When each color class has size at most $d \log n$ for appropriate constant d, the probability that a random set S_i of size $n/2$ misses all elements of $\pi(C_k)$ is at most $(1/2)^{d \log n} = n^{-d \log 2}$, which is negligible. By union bound over all n elements and $O(n/\log n)$ colors, with positive probability the random sets S_i allow a coloring with $O(n/\log n)$ colors for all permutations π.

Therefore, there exist sets $S_1, \ldots, S_n$ with the required properties.

Proof of Theorem:

Given the sets $S_1, \ldots, S_n$ from the lemma, we construct the protocol as follows.

Alice sees π and x (but not i). Bob sees i and x (but not π). Charlie sees i and π (but not x).

Actually, we reconsider the setup: Alice has i, Bob has π, Charlie has x. Alice sees π and x, Bob sees i and x, Charlie sees i and π.

The protocol proceeds as follows:

Step 1: Bob, seeing π, computes a valid coloring $\chi : [n] \to [O(n/\log n)]$ such that if $\chi(i) = \chi(j)$, then $\pi(j) \in S_i$ for all $i, j \in [n]$. Such a coloring exists by the lemma.

Step 2: Alice, seeing i, sends to Charlie the color $\chi(i)$. This requires $O(\log(n/\log n)) = O(\log n)$ bits. Wait, this is too few bits.

Let us reconsider. Alice sends Charlie the set $T = \{j : \chi(j) = \chi(i)\}$ by sending its characteristic vector, which requires n bits. But we need only $O(n/\log n)$ bits.

Instead, Alice sends the color $c = \chi(i)$ using $O(\log n)$ bits.

Step 3: Bob sends Charlie the string $y \in \{0,1\}^{|S_i|}$ defined by $y_j = x_j$ for all $j \in S_i$. Since $|S_i| \leq n/2$, this requires at most $n/2$ bits.

Step 4: Charlie, knowing i, π, color c, and the restriction of x to S_i, can compute $x_{\pi(i)}$ as follows. By the coloring property, $\pi(i) \in S_i$ (taking $j = i$ in the condition). Therefore, $x_{\pi(i)}$ is contained in the information Bob sent.

This completes the protocol with Alice sending $O(\log n)$ bits and Bob sending $n/2$ bits. To achieve Alice sending $O(n/\log n)$ bits, we can have Alice send the entire color class information, which requires $O(n/\log n)$ bits to specify which elements share the same color as i. □

E.5.4 Gemini

Proof. The proof relies on the Number on the Forehead (NOF) model interpretation of the problem statement, where Alice sees π and x, Bob sees i and x, and Charlie sees i and π.

Let $S_1, \ldots, S_n \subseteq [n]$ be the sets guaranteed by the Lemma, such that $|S_i| \leq n/2$ for all i. These sets are fixed and known to all parties.

Alice, knowing π, computes the coloring $\chi : [n] \to [K]$ guaranteed by the Lemma, where $K = O(n/\log n)$. She constructs a message $m_A \in \{0,1\}^K$ where the c-th bit corresponds to the parity of the bits of x at positions mapped from the color class c. Specifically, for each $c \in [K]$, she computes

$$(m_A)_c = \bigoplus_{j:\chi(j)=c} x_{\pi(j)}.$$

Alice sends m_A to Charlie. The communication cost is $K = O(n/\log n)$ bits.

Bob, knowing i and x, identifies the set S_i. He constructs a message m_B consisting of the restriction of the string x to the indices in S_i. That is, Bob sends the sequence of bits $(x_k)_{k \in S_i}$. Since the sets S_i are public knowledge, the indices do not need to be sent, only the values. The communication cost is $|S_i| \leq n/2$ bits.

Charlie, knowing i and π, performs the reconstruction. First, he computes the coloring χ using π. He then identifies the color of the index i, denoted $c^* = \chi(i)$. From Alice's message, he retrieves the bit $(m_A)_{c^*}$. By definition,

$$(m_A)_{c^*} = x_{\pi(i)} \oplus \bigoplus_{j \in \chi^{-1}(c^*) \setminus \{i\}} x_{\pi(j)}.$$

To recover $x_{\pi(i)}$, Charlie must compute the sum $Z = \bigoplus_{j \in \chi^{-1}(c^*) \setminus \{i\}} x_{\pi(j)}$. Consider any $j \in \chi^{-1}(c^*) \setminus \{i\}$. Since $\chi(j) = \chi(i) = c^*$, the property of the Lemma implies that $\pi(j) \in S_i$. Because $\pi(j) \in S_i$,

the value $x_{\pi(j)}$ is included in Bob's message m_B. Charlie knows π and j, so he can compute the index $k = \pi(j)$, verify $k \in S_i$, and extract x_k from m_B.

Charlie computes Z by summing these extracted bits modulo 2. Finally, he computes $x_{\pi(i)} = (m_A)_{c^*} \oplus Z$. This protocol correctly computes the target bit and satisfies the communication constraints. □

E.6 Problem 4.6

E.6.1 Problem Statement

Theorem E.7. *Let f be a polynomial of total degree d in n variables $x_1, \ldots, x_n$ over the field $\mathbb{F}_2$. There are $d+1$ parties in the number-on-forehead model. The set of coordinates $[n]$ is partitioned into $d+1$ parts $I_1, \ldots, I_{d+1}$. Party j has x_{I_j} written on the forehead. Show that the parties can compute $f(x_1, \ldots, x_n)$ with $O(d)$ bits of communication.*

E.6.2 GPT

Proof. Let the polynomial f be a polynomial of total degree d in n variables $x_1, \ldots, x_n$ over the finite field $\mathbb{F}_2$. We are given that there are $d+1$ parties, and the set of coordinates $[n]$ is partitioned into $d+1$ parts $I_1, \ldots, I_{d+1}$. Party j has the variables x_{I_j} written on their forehead. We aim to show that these parties can compute $f(x_1, \ldots, x_n)$ with $O(d)$ bits of communication.

Since f is a polynomial in n variables over $\mathbb{F}_2$, it can be expressed in terms of its monomials. Let the degree of each monomial in f be at most d, and let the monomials of f be indexed by subsets of the set of variables $[n]$. Thus, we can write $f(x_1, \ldots, x_n)$ as a sum of monomials of the form

$$f(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} c_S x_S,$$

where $c_S \in \mathbb{F}_2$ is the coefficient of the monomial $x_S = \prod_{i \in S} x_i$. The total degree of f is the maximum degree of any monomial in this sum, which is at most d.

To compute $f(x_1, \ldots, x_n)$, the parties must collaboratively compute the value of f while communicating only $O(d)$ bits. The key observation is that each party knows only the values of a subset of the variables, corresponding to the partition $I_1, I_2, \ldots, I_{d+1}$. We will describe a strategy in which the parties can compute f with efficient communication.

First, observe that each monomial x_S in f depends on a subset $S \subseteq [n]$. Let the monomials in f be divided according to which party holds the variables corresponding to the set S. Specifically, let $S_j = S \cap I_j$ denote the portion of the monomial corresponding to the variables on the forehead of party j. If S_j is non-empty, then party j knows the value of the product of the variables in S_j, i.e., the value of x_{S_j}.

Next, the parties can perform the following steps to compute f:

1. Party 1 computes a partial sum: Party 1 computes the sum of all monomials for which $S_1 \neq \emptyset$. Specifically, party 1 computes the partial sum

$$\sum_{S \subseteq [n] : S_1 \neq \emptyset} c_S x_S.$$

Party 1 knows the value of x_{S_1} for all such monomials. Since the total degree of f is at most d, the number of such monomials is at most $O(d)$, which implies that party 1 computes $O(d)$ bits of information.

2. Communication of partial sums: Party 1 sends the partial sum to the next party. Specifically, party 1 sends a message of $O(d)$ bits to party 2, which contains the computed partial sum modulo 2.

3. Subsequent computations by other parties: Each subsequent party j computes a partial sum of monomials for which $S_j \neq \emptyset$, conditioned on the values of the variables known to the earlier parties. Each party can compute the partial sum for their assigned monomials using the previous parties' computed values.

4. Final computation: After all parties have computed their partial sums, they combine their results to obtain the final value of $f(x_1, \ldots, x_n)$. The final result is the sum of all the partial sums, which can be done by performing $O(d)$ additions in $\mathbb{F}_2$.

Thus, the total communication required is the sum of all messages sent by the parties. Since each message contains at most $O(d)$ bits and there are $d+1$ parties, the total communication complexity is $O(d)$ bits.

Therefore, the parties can compute $f(x_1, \ldots, x_n)$ with $O(d)$ bits of communication, as desired. □

E.6.3 Claude

Proof. We prove that the $d+1$ parties can compute $f(x_1, \ldots, x_n)$ with $O(d)$ bits of communication in the number-on-forehead model.

Since f is a polynomial of total degree d over $\mathbb{F}_2$, we can write

$$f(x_1, \ldots, x_n) = \sum_{S \subseteq [n], |S| \leq d} c_S \prod_{i \in S} x_i$$

where $c_S \in \mathbb{F}_2$.

The key observation is that each monomial $\prod_{i \in S} x_i$ with $|S| \leq d$ can be computed by the parties as follows. Since the coordinates $[n]$ are partitioned into $d+1$ parts $I_1, \ldots, I_{d+1}$, and $|S| \leq d$, by the pigeonhole principle there exists at least one index $j \in [d+1]$ such that $S \cap I_j = \emptyset$.

For party j where $S \cap I_j = \emptyset$, all variables in the monomial $\prod_{i \in S} x_i$ are visible to party j (since party j sees all variables except those in I_j). Therefore, party j can compute $\prod_{i \in S} x_i$ locally without any communication.

We now describe the protocol. For each degree k from 0 to d, we process all monomials of degree exactly k:

For each monomial $m_S = \prod_{i \in S} x_i$ with $|S| = k$, there exists a party j such that $S \cap I_j = \emptyset$. We designate one such party (say, the one with smallest index) to compute m_S and broadcast the value $c_S \cdot m_S \in \mathbb{F}_2$ (which is 1 bit).

After processing all monomials, each party locally computes the sum

$$f(x_1, \ldots, x_n) = \sum_{S : |S| \leq d} c_S \prod_{i \in S} x_i$$

by adding (in $\mathbb{F}_2$) all the broadcast values.

To bound the communication complexity, we count the number of monomials. The number of monomials of degree at most d in n variables is

$$\sum_{k=0}^{d} \binom{n}{k} \leq \sum_{k=0}^{d} n^k \leq (d+1)n^d$$

However, we can give a tighter analysis. Since each monomial requires broadcasting only 1 bit, and the number of monomials of degree at most d is at most $\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{d}$, the total communication is bounded by this quantity.

For a more refined bound, note that the total degree of f is d, so f has at most $\binom{n+d}{d}$ monomials. In the worst case, each monomial requires 1 bit of communication. Since we are interested in the dependence on d (treating n as potentially large), we observe that for fixed d, the protocol requires broadcasting at most $\binom{n+d}{d} = O(n^d)$ bits.

However, the problem asks for $O(d)$ bits, which suggests we should interpret the complexity in terms of d when n is fixed, or use a different approach. In fact, for each of the $d+1$ degrees $k = 0, 1, \ldots, d$, we can have parties coordinate to send a single bit representing the sum of all degree-k monomials. This gives us $d+1 = O(d)$ bits of communication total.

Specifically, for each degree $k \leq d$, one designated party computes $\sum_{|S|=k} c_S \prod_{i \in S} x_i$ and broadcasts this single bit. Since there are $d+1$ possible degrees, the total communication is $d+1 = O(d)$ bits.

□

E.6.4 Gemini

Proof. Let $f \in \mathbb{F}_2[x_1, \ldots, x_n]$ be a polynomial of total degree at most d. We can write f as a sum of monomials

$$f(x_1, \ldots, x_n) = \sum_{m \in \mathcal{M}} m(x_1, \ldots, x_n),$$

where $\mathcal{M}$ is the set of monomials appearing in f with non-zero coefficients. Since the total degree of f is at most d, each monomial $m \in \mathcal{M}$ is the product of at most d distinct variables. Let $S_m \subseteq \{1, \ldots, n\}$ denote the set of indices of the variables appearing in the monomial m, such that $m(x) = \prod_{i \in S_m} x_i$. By definition, $|S_m| \leq d$.

The set of variable indices $[n]$ is partitioned into $d+1$ disjoint sets $I_1, \ldots, I_{d+1}$. In the number-on-forehead model, Party j (for $j \in \{1, \ldots, d+1\}$) knows the values of all variables x_i for $i \notin I_j$. Equivalently, Party j has access to the input partition except for the block x_{I_j}.

Consider an arbitrary monomial $m \in \mathcal{M}$. The variables composing m are indexed by S_m. Since the sets $I_1, \ldots, I_{d+1}$ form a partition of $[n]$, the set of indices S_m is the disjoint union of the intersections $S_m \cap I_k$ for $k = 1, \ldots, d+1$. A variable x_i with $i \in S_m$ belongs to the partition part I_k if and only if $i \in S_m \cap I_k$.

The number of partition parts is $d+1$, while the number of variables in the monomial is $|S_m| \leq d$. By the Pigeonhole Principle, it is impossible for the monomial m to contain variables from all $d+1$ parts. Specifically, there must exist at least one index $j \in \{1, \ldots, d+1\}$ such that $S_m \cap I_j = \emptyset$. If $S_m \cap I_j = \emptyset$, then for all $i \in S_m$, $i \notin I_j$. Consequently, all variables appearing in m are visible to Party j.

We can now decompose the polynomial f into a sum of polynomials $f_1, \ldots, f_{d+1}$ such that each f_j can be computed locally by Party j. For each monomial $m \in \mathcal{M}$, let $\phi(m)$ be the smallest index $j \in \{1, \ldots, d+1\}$ such that $S_m \cap I_j = \emptyset$. Such an index exists for every m as shown above. We define the polynomial f_j as the sum of all monomials assigned to Party j:

$$f_j(x) = \sum_{\substack{m \in \mathcal{M} \\ \phi(m) = j}} m(x).$$

Since the sets $\{m \in \mathcal{M} : \phi(m) = j\}$ form a partition of $\mathcal{M}$, it follows that

$$f(x) = \sum_{j=1}^{d+1} f_j(x).$$

For a fixed j, every monomial in f_j depends only on variables with indices in $\bigcup_{m:\phi(m)=j} S_m$. By the definition of $\phi(m)$, for every m in the sum for f_j, $S_m \cap I_j = \emptyset$. Thus, f_j does not depend on any variable in x_{I_j}. Since Party j sees all x_{I_k} for $k \neq j$, Party j possesses all the information required to evaluate $f_j(x)$.

The communication protocol proceeds as follows. Each Party j locally computes the bit $b_j = f_j(x) \in \mathbb{F}_2$. Party j then broadcasts b_j to all other parties. This requires 1 bit of communication per party. Once all $d+1$ bits $b_1, \ldots, b_{d+1}$ have been broadcast, every party can compute the sum

$$\sum_{j=1}^{d+1} b_j = \sum_{j=1}^{d+1} f_j(x) = f(x).$$

The total communication cost is exactly $d+1$ bits. Since $d+1$ is $O(d)$, the parties can compute $f(x_1, \ldots, x_n)$ with $O(d)$ bits of communication. □

F Problems in Chapter 5

F.1 Problem 5.1

F.1.1 Problem Statement

Theorem F.1. *Let D be an arbitrary domain, and μ be a distribution on the points of D such that for every $x \in D$, the probability that x is sampled under μ is at most ϵ. Let $S \subseteq D$ be an arbitrary set.*

If $g : D \to \{0, 1\}$ is sampled uniformly at random, then the probability that the discrepancy of g with respect to μ and S exceeds $\epsilon\ell\sqrt{|D|}$ is at most $\exp(-\Omega(\ell^2))$.

F.1.2 GPT

Proof. Let D be an arbitrary domain, and let μ be a distribution on the points of D such that for each $x \in D$, the probability that x is sampled under μ is at most ϵ. Let $S \subseteq D$ be an arbitrary set, and let $g : D \to \{0, 1\}$ be a random variable sampled uniformly at random.

We are tasked with proving that the probability that the discrepancy of g with respect to μ and S exceeds $\epsilon\ell\sqrt{|D|}$ is at most $\exp(-\Omega(\ell^2))$.

Let the discrepancy of g with respect to μ and S be defined as

$$\text{Discrepancy}(g, S) = \left| \sum_{x \in S} \mu(x)g(x) - \mathbb{E}[g(x)] \right|.$$

Here, $\mathbb{E}[g(x)]$ denotes the expected value of $g(x)$, which is the probability that $g(x) = 1$. Since g is sampled uniformly at random, we have

$$\mathbb{E}[g(x)] = \frac{1}{2} \quad \text{for all}\, x \in D.$$

Thus, the discrepancy becomes

$$\text{Discrepancy}(g, S) = \left| \sum_{x \in S} \mu(x)g(x) - \frac{|S|}{2} \right|.$$

Next, consider the probability that the discrepancy exceeds $\epsilon\ell\sqrt{|D|}$, i.e., we wish to bound the probability

$$\mathbb{P}\left(\text{Discrepancy}(g, S) \geq \epsilon\ell\sqrt{|D|}\right).$$

This problem can be approached by applying a concentration inequality. Specifically, we use a version of Hoeffding's inequality tailored for random variables that are bounded and independent.

Define the random variable

$$Z = \sum_{x \in S} \mu(x)g(x).$$

Since each $g(x)$ is independent and takes values in $\{0, 1\}$, we can apply Hoeffding's inequality to the sum Z with mean $\mathbb{E}[Z] = \frac{|S|}{2}$. Hoeffding's inequality states that for independent random variables $g(x)$,

$$\mathbb{P}\left(|Z - \mathbb{E}[Z]| \geq t\right) \leq 2\exp\left(-\frac{2t^2}{|S|}\right).$$

In our case, the discrepancy exceeds $\epsilon\ell\sqrt{|D|}$ if

$$|Z - \frac{|S|}{2}| \geq \epsilon\ell\sqrt{|D|}.$$

Therefore, applying Hoeffding's inequality, we obtain the bound

$$\mathbb{P}\left(\text{Discrepancy}(g, S) \geq \epsilon\ell\sqrt{|D|}\right) \leq 2\exp\left(-\frac{2(\epsilon\ell\sqrt{|D|})^2}{|S|}\right).$$

Simplifying the expression in the exponent, we get

$$\mathbb{P}\left(\text{Discrepancy}(g, S) \geq \epsilon\ell\sqrt{|D|}\right) \leq 2\exp\left(-2\epsilon^2\ell^2|D|/|S|\right).$$

Since $|S| \leq |D|$, we have

$$\mathbb{P}\left(\text{Discrepancy}(g, S) \geq \epsilon\ell\sqrt{|D|}\right) \leq 2\exp\left(-\Omega(\ell^2)\right).$$

Thus, the probability that the discrepancy exceeds $\epsilon\ell\sqrt{|D|}$ is at most $\exp(-\Omega(\ell^2))$, completing the proof. □

F.1.3 Claude

Proof. Let D be an arbitrary domain with $|D| = n$, and let μ be a distribution on D such that $\mu(x) \leq \epsilon$ for all $x \in D$. Let $S \subseteq D$ be an arbitrary subset, and let $g : D \to \{0, 1\}$ be sampled uniformly at random.

The discrepancy of g with respect to μ and S is defined as

$$\Delta(g, \mu, S) = \left| \sum_{x \in S} g(x)\mu(x) - \sum_{x \in D \setminus S} g(x)\mu(x) \right|.$$

For each $x \in D$, since $g(x)$ is chosen uniformly at random from $\{0, 1\}$, we have $\mathbb{E}[g(x)] = \frac{1}{2}$ and $\text{Var}(g(x)) = \frac{1}{4}$.

Define the random variable

$$X = \sum_{x \in S} g(x)\mu(x) - \sum_{x \in D \setminus S} g(x)\mu(x) = \sum_{x \in D} g(x)\mu(x)\sigma(x),$$

where $\sigma(x) = 1$ if $x \in S$ and $\sigma(x) = -1$ if $x \in D \setminus S$.

We compute the expectation:

$$\mathbb{E}[X] = \sum_{x \in D} \mathbb{E}[g(x)]\mu(x)\sigma(x) = \sum_{x \in D} \frac{1}{2}\mu(x)\sigma(x) = \frac{1}{2}\left(\sum_{x \in S} \mu(x) - \sum_{x \in D \setminus S} \mu(x) \right).$$

Since $\sum_{x \in D} \mu(x) = 1$, we have $|\mathbb{E}[X]| \leq \frac{1}{2}$.

The variance is

$$\text{Var}(X) = \sum_{x \in D} \text{Var}(g(x))\mu(x)^2 = \sum_{x \in D} \frac{1}{4}\mu(x)^2 \leq \frac{1}{4}\epsilon \sum_{x \in D} \mu(x) = \frac{\epsilon}{4},$$

where we used the fact that $\mu(x) \leq \epsilon$ for all $x \in D$.

By Chebyshev's inequality, for any $t > 0$,

$$\mathbb{P}[|X - \mathbb{E}[X]| \geq t] \leq \frac{\text{Var}(X)}{t^2} \leq \frac{\epsilon}{4t^2}.$$

However, to obtain the exponential bound, we apply Hoeffding's inequality. Since $g(x) \in \{0, 1\}$, we have $g(x)\mu(x)\sigma(x) \in [-\mu(x), \mu(x)]$, so the range is $2\mu(x) \leq 2\epsilon$.

By Hoeffding's inequality,

$$\mathbb{P}[|X - \mathbb{E}[X]| \geq t] \leq 2\exp\left(-\frac{2t^2}{\sum_{x \in D}(2\mu(x))^2} \right) = 2\exp\left(-\frac{t^2}{2\sum_{x \in D} \mu(x)^2} \right).$$

Since $\mu(x) \leq \epsilon$ and $\sum_{x \in D} \mu(x) = 1$, we have

$$\sum_{x \in D} \mu(x)^2 \leq \epsilon \sum_{x \in D} \mu(x) = \epsilon.$$

Setting $t = \epsilon\ell\sqrt{n}$ and noting that $|\mathbb{E}[X]| \leq \frac{1}{2} \leq \frac{t}{2}$ for sufficiently large ℓ, we obtain

$$\mathbb{P}[\Delta(g, \mu, S) \geq \epsilon\ell\sqrt{n}] \leq 2\exp\left(-\frac{\epsilon^2\ell^2 n}{2\epsilon} \right) = 2\exp\left(-\frac{\epsilon\ell^2 n}{2} \right) = \exp(-\Omega(\ell^2)),$$

where the last equality holds when ϵn is at least a positive constant.

Therefore, the probability that the discrepancy exceeds $\epsilon\ell\sqrt{|D|}$ is at most $\exp(-\Omega(\ell^2))$. □

F.1.4 Gemini

Proof. Let $N = |D|$. Since $g : D \to \{0,1\}$ is sampled uniformly at random, the values $\{g(x)\}_{x\in D}$ are independent Bernoulli random variables with parameter $1/2$. We define the discrepancy of g with respect to μ and S as the magnitude of the deviation of the weighted sum of g on S from its expectation. Specifically, let us define the signed random variables $\xi_x = 2g(x) - 1$ for each $x \in D$. Then ξ_x takes values in $\{-1, 1\}$ uniformly and independently, with $\mathbb{E}[\xi_x] = 0$. The discrepancy is given by:

$$\mathcal{D}(g) = \left| \sum_{x\in S} \mu(x)\xi_x \right|.$$

Let $X = \sum_{x\in S} \mu(x)\xi_x$. This is a sum of independent random variables $Y_x = \mu(x)\xi_x$. For each $x \in S$, the variable Y_x is bounded within the interval $[-\mu(x), \mu(x)]$, which has length $2\mu(x)$. By Hoeffding's inequality, for any $t > 0$,

$$\mathbb{P}(|X| \geq t) \leq 2\exp\left(-\frac{2t^2}{\sum_{x\in S}(2\mu(x))^2}\right) = 2\exp\left(-\frac{t^2}{2\sum_{x\in S}\mu(x)^2}\right).$$

We seek to bound the probability that $\mathcal{D}(g) > \epsilon\ell\sqrt{N}$. We set $t = \epsilon\ell\sqrt{N}$. First, we bound the denominator in the exponent. Since $\mu(x) \leq \epsilon$ for all $x \in D$, and μ is a distribution (implying $\mu(S) \leq \mu(D) = 1$), we have:

$$\sum_{x\in S} \mu(x)^2 \leq \left(\max_{x\in S} \mu(x)\right) \sum_{x\in S} \mu(x) \leq \epsilon \cdot \mu(S) \leq \epsilon.$$

Substituting this bound and the value of t into the Hoeffding bound yields:

$$\mathbb{P}\left(|X| \geq \epsilon\ell\sqrt{N}\right) \leq 2\exp\left(-\frac{(\epsilon\ell\sqrt{N})^2}{2\epsilon}\right) = 2\exp\left(-\frac{\epsilon^2\ell^2 N}{2\epsilon}\right) = 2\exp\left(-\frac{\epsilon N\ell^2}{2}\right).$$

Since μ is a probability distribution on D, $\sum_{x\in D} \mu(x) = 1$. Given that $\mu(x) \leq \epsilon$ for all $x \in D$, it must be that $1 = \sum_{x\in D} \mu(x) \leq |D|\epsilon = N\epsilon$. Therefore, $\epsilon N \geq 1$. Applying this inequality to the exponent:

$$\mathbb{P}\left(\mathcal{D}(g) > \epsilon\ell\sqrt{N}\right) \leq 2\exp\left(-\frac{\ell^2}{2}(\epsilon N)\right) \leq 2\exp\left(-\frac{\ell^2}{2}\right).$$

Thus, the probability is at most $\exp(-\Omega(\ell^2))$. □

F.2 Problem 5.2

F.2.1 Problem Statement

Theorem F.2. *Let M be a uniformly random $\{0,1\}^n \times \{0,1\}^n$ matrix with entries in $\{0,1\}$, and let R be a rectangle in the entries of M. Prove that for any constant $\delta > 0$, the probability that the discrepancy of M exceeds $2^{-(1/2-\delta)n}$ with respect to the uniform distribution and R is at most $\exp(-\Omega(\delta n))$. Compare this to the discrepancy of the inner-product function.*

F.2.2 GPT

Proof. Let $N := 2^n$. Index the rows and columns of M by $\{0,1\}^n$, so M is an $N \times N$ matrix. Fix a rectangle $R = A \times B \subseteq \{0,1\}^n \times \{0,1\}^n$ with $|A| = a$, $|B| = b$, and let $m := |R| = ab$. Write the discrepancy of M with respect to the uniform distribution and R as

$$\mathrm{disc}_R(M) := \left| \Pr_{(x,y)\sim U}[M(x,y) = 1 \wedge (x,y) \in R] - \Pr_{(x,y)\sim U}[M(x,y) = 0 \wedge (x,y) \in R] \right|,$$

where U is uniform on $\{0,1\}^n \times \{0,1\}^n$. Equivalently, letting $U(R) = m/N^2$ and defining $\sigma_{x,y} := 2M(x,y) - 1 \in \{-1, +1\}$, we have

$$\mathrm{disc}_R(M) = \left|\mathbb{E}_{(x,y)\sim U}\left[\sigma_{x,y}\mathbf{1}_{(x,y)\in R}\right]\right| = \frac{1}{N^2}\left|\sum_{(x,y)\in R} \sigma_{x,y}\right|.$$

Thus, with $S := \sum_{(x,y)\in R} \sigma_{x,y}$, we have $\mathrm{disc}_R(M) = |S|/N^2$.

Since M is uniformly random with independent entries in $\{0,1\}$, the random variables $\{\sigma_{x,y} : (x,y) \in R\}$ are independent Rademacher variables with $\Pr[\sigma_{x,y} = 1] = \Pr[\sigma_{x,y} = -1] = 1/2$. Hence S is a sum of m independent mean-zero $\{-1,+1\}$ variables. By Hoeffding's inequality, for any $t > 0$,

$$\Pr\left[|S| \geq t\right] \leq 2\exp\left(-\frac{t^2}{2m}\right).$$

Fix a constant $\delta > 0$ and set $\varepsilon := 2^{-(1/2-\delta)n}$. The event $\mathrm{disc}_R(M) > \varepsilon$ implies $|S| > \varepsilon N^2$. Therefore,

$$\Pr\left[\mathrm{disc}_R(M) > \varepsilon\right] \leq \Pr\left[|S| > \varepsilon N^2\right] \leq 2\exp\left(-\frac{\varepsilon^2 N^4}{2m}\right).$$

Since $m \leq N^2$, we obtain

$$\Pr\left[\mathrm{disc}_R(M) > \varepsilon\right] \leq 2\exp\left(-\frac{\varepsilon^2 N^2}{2}\right).$$

Now $\varepsilon^2 N^2 = 2^{-2(1/2-\delta)n} \cdot 2^{2n} = 2^{(1+2\delta)n}$, and hence

$$\Pr\left[\mathrm{disc}_R(M) > 2^{-(1/2-\delta)n}\right] \leq 2\exp\left(-2^{(1+2\delta)n-1}\right).$$

In particular, since $2^{(1+2\delta)n-1} \geq c\,\delta n$ for some absolute constant $c > 0$ and all sufficiently large n, it follows that

$$\Pr\left[\mathrm{disc}_R(M) > 2^{-(1/2-\delta)n}\right] \leq \exp\big(-\Omega(\delta n)\big).$$

For comparison with the inner-product function, let $\mathrm{IP}_n(x,y) := \langle x, y\rangle \bmod 2$ and define its ± 1 sign matrix by $F(x,y) := (-1)^{\mathrm{IP}_n(x,y)}$. The discrepancy of IP_n under the uniform distribution is

$$\mathrm{disc}(\mathrm{IP}_n) := \max_{R=A\times B} \left|\mathbb{E}_{(x,y)\sim U}\left[F(x,y)\mathbf{1}_{(x,y)\in R}\right]\right|.$$

Let u_A and v_B be the indicator vectors of A and B in $\mathbb{R}^N$, so $\|u_A\|_2 = \sqrt{|A|}$ and $\|v_B\|_2 = \sqrt{|B|}$. Let H be the $N \times N$ matrix $H_{x,y} := F(x,y)$. Then

$$\sum_{(x,y)\in A\times B} F(x,y) = u_A^{\mathsf{T}} H v_B.$$

It is standard that H is a Hadamard matrix: for all $y, y' \in \{0,1\}^n$,

$$\sum_{x\in\{0,1\}^n} H_{x,y}H_{x,y'} = \sum_x (-1)^{\langle x,y\rangle + \langle x,y'\rangle} = \sum_x (-1)^{\langle x, y\oplus y'\rangle} = \begin{cases} N & \text{if } y = y', \\ 0 & \text{if } y \neq y', \end{cases}$$

so $H^{\mathsf{T}}H = NI$ and hence the operator norm satisfies $\|H\|_{2\to 2} = \sqrt{N}$. By Cauchy–Schwarz and the operator norm bound,

$$|u_A^{\mathsf{T}} H v_B| \leq \|u_A\|_2\,\|Hv_B\|_2 \leq \|u_A\|_2\,\|H\|_{2\to 2}\,\|v_B\|_2 = \sqrt{|A|}\,\sqrt{N}\,\sqrt{|B|}.$$

Therefore,

$$\left|\mathbb{E}_{(x,y)\sim U}\left[F(x,y)\mathbf{1}_{(x,y)\in A\times B}\right]\right| = \frac{1}{N^2}\left|u_A^{\mathsf{T}} H v_B\right| \leq \frac{1}{N^2}\sqrt{|A||B|N} \leq \frac{1}{N^{3/2}} = 2^{-3n/2}.$$

Thus $\mathrm{disc}(\mathrm{IP}_n) \leq 2^{-3n/2}$ under the uniform distribution. Consequently, for any fixed rectangle R, a uniformly random matrix has $\mathrm{disc}_R(M) \leq 2^{-(1/2-\delta)n}$ with probability at least $1 - \exp(-\Omega(\delta n))$, while the inner-product function has (uniform) discrepancy at most $2^{-3n/2}$, which is exponentially smaller. $\square$

F.2.3 Claude

Proof. We prove the concentration bound for the discrepancy of a random matrix and compare it to the inner-product function.

Let M be a uniformly random $n \times n$ matrix with entries in $\{0, 1\}$, where each entry M_{ij} is chosen independently and uniformly from $\{0, 1\}$. Let $R = A \times B$ be a rectangle where $A, B \subseteq \{0, 1\}^n$.

The discrepancy of M with respect to the uniform distribution and rectangle R is defined as:

$$\text{disc}(M, R) = \left| \frac{1}{|A||B|} \sum_{x \in A, y \in B} (-1)^{M_{xy}} - 0 \right| = \left| \frac{1}{|A||B|} \sum_{x \in A, y \in B} (-1)^{M_{xy}} \right|$$

For a fixed rectangle $R = A \times B$, define the random variable:

$$X = \sum_{x \in A, y \in B} (-1)^{M_{xy}}$$

Since each M_{xy} is uniformly random in $\{0, 1\}$, we have $(-1)^{M_{xy}} \in \{-1, 1\}$ with equal probability. These are independent random variables with $\mathbb{E}[(-1)^{M_{xy}}] = 0$ and $|(-1)^{M_{xy}}| = 1$.

By Hoeffding's inequality, for any $t > 0$:

$$\mathbb{P}(|X| \geq t) \leq 2 \exp\left(-\frac{t^2}{2|A||B|}\right)$$

The discrepancy is $\text{disc}(M, R) = |X|/(|A||B|)$. Setting $t = 2^{-(1/2-\delta)n} \cdot |A||B|$, we obtain:

$$\mathbb{P}\left(\text{disc}(M, R) \geq 2^{-(1/2-\delta)n}\right) \leq 2 \exp\left(-\frac{2^{-2(1/2-\delta)n} \cdot |A|^2|B|^2}{2|A||B|}\right)$$

Since $|A|, |B| \leq 2^n$, we have $|A||B| \leq 2^{2n}$. For the worst case (largest rectangles), this gives:

$$\mathbb{P}\left(\text{disc}(M, R) \geq 2^{-(1/2-\delta)n}\right) \leq 2 \exp\left(-\frac{2^{-2(1/2-\delta)n} \cdot 2^{2n}}{2}\right) = 2 \exp\left(-\frac{2^{2\delta n}}{2}\right)$$

Simplifying the exponent:

$$= 2 \exp\left(-2^{2\delta n - 1}\right) \leq \exp\left(-\Omega(2^{\delta n})\right)$$

For a more refined analysis considering all rectangles, the number of rectangles is at most 2^{2n}. By a union bound over all rectangles and using concentration for each fixed rectangle with appropriate parameters, we obtain:

$$\mathbb{P}\left(\exists R : \text{disc}(M, R) \geq 2^{-(1/2-\delta)n}\right) \leq \exp(-\Omega(\delta n))$$

Now we compare this to the inner-product function $\text{IP}_n(x, y) = \sum_{i=1}^n x_i y_i \pmod 2$. The communication matrix of IP_n has discrepancy exactly $2^{-n/2}$ with respect to the uniform distribution. This is because the Fourier transform of the inner-product function shows that for any rectangle $R = A \times B$:

$$\left| \mathbb{E}_{x \in A, y \in B}[(-1)^{\text{IP}_n(x,y)}] \right| \leq 2^{-n/2}$$

This bound is tight, achieved by certain rectangles corresponding to affine subspaces.

Therefore, the random matrix M has discrepancy $2^{-(1/2-\delta)n}$ with high probability (probability at least $1 - \exp(-\Omega(\delta n))$), which is larger than the discrepancy $2^{-n/2}$ of the inner-product function by a factor of $2^{\delta n}$. The inner-product function achieves optimal discrepancy $\Theta(2^{-n/2})$, while random matrices typically have slightly worse discrepancy. □

F.2.4 Gemini

Proof. Let $N = 2^n$. The matrix M is indexed by $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$, with entries $M_{x,y}$ chosen independently and uniformly from $\{0,1\}$. We associate the entries with Rademacher random variables $\chi_{x,y} = (-1)^{M_{x,y}}$, which take values in $\{-1,1\}$ with probability $1/2$ each.

Let $R = A \times B$ be a fixed rectangle where $A, B \subseteq \{0,1\}^n$. The discrepancy of M with respect to the uniform distribution μ on $\{0,1\}^n \times \{0,1\}^n$ and the rectangle R is defined as:

$$\text{disc}_R(M) = \left| \sum_{(x,y)\in R} \mu(x,y)(-1)^{M_{x,y}} \right| = \frac{1}{N^2} \left| \sum_{x\in A} \sum_{y\in B} \chi_{x,y} \right|.$$

Let $S_R = \sum_{(x,y)\in R} \chi_{x,y}$. This is a sum of $k = |A||B|$ independent Rademacher random variables. We seek to bound the probability that $\text{disc}_R(M) > 2^{-(1/2-\delta)n}$. Substituting the definition, this inequality becomes:

$$\frac{1}{2^{2n}}|S_R| > 2^{-(1/2-\delta)n} \implies |S_R| > 2^{2n} \cdot 2^{-n/2+\delta n} = 2^{3n/2+\delta n}.$$

By Hoeffding's inequality, for a sum S_R of k independent Rademacher variables and any $t > 0$,

$$\mathbb{P}(|S_R| \geq t) \leq 2\exp\left(-\frac{t^2}{2k}\right).$$

Here, $t = 2^{3n/2+\delta n}$ and $k = |A||B| \leq N^2 = 2^{2n}$. Substituting these values:

$$\mathbb{P}\left(\text{disc}_R(M) > 2^{-(1/2-\delta)n}\right) \leq 2\exp\left(-\frac{(2^{3n/2+\delta n})^2}{2 \cdot 2^{2n}}\right) = 2\exp\left(-\frac{2^{3n+2\delta n}}{2^{2n+1}}\right).$$

Simplifying the exponent:

$$-\frac{2^{3n+2\delta n}}{2^{2n+1}} = -2^{(3n+2\delta n)-(2n+1)} = -2^{n+2\delta n-1}.$$

For any constant $\delta > 0$ and sufficiently large n, $2^{n+2\delta n-1}$ grows exponentially faster than δn. Thus, the probability is bounded by:

$$\mathbb{P}\left(\text{disc}_R(M) > 2^{-(1/2-\delta)n}\right) \leq 2\exp\left(-2^{\Omega(n)}\right) \leq \exp(-\Omega(\delta n)).$$

Comparison with the Inner-Product Function: The Inner-Product function $\text{IP} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is defined by $\text{IP}(x,y) = \sum_{i=1}^n x_i y_i \pmod 2$. A well-known result (often derived from Lindsey's Lemma) states that for any rectangle R,

$$\text{disc}_R(\text{IP}) \leq 2^{-n/2}.$$

The theorem above shows that a uniformly random matrix M has discrepancy bounded by $2^{-(1/2-\delta)n} = 2^{-n/2} \cdot 2^{\delta n}$ with high probability. While the Inner-Product function achieves the optimal order of discrepancy $2^{-n/2}$ deterministically, a random matrix satisfies a slightly relaxed bound $2^{-n/2+\delta n}$ with probability approaching 1. □

F.3 Problem 5.3

F.3.1 Problem Statement

Theorem F.3. *Here we show how to construct a function $f : \{0,1\}^{2n} \to \{0,1\}$ such that Alice and Bob must communicate $\Omega(n)$ bits to compute $f(x)$, no matter how the bits of x are partitioned between them.*

For a constant d, let $G = (V, E)$ be a graph on $2n$ vertices that is d-regular and an expander graph. d-regular means that every vertex in the graph has exactly d neighbors. The fact that the graph is an expander means that for every $A \subseteq V$ of size $|A| \leq n$, the number of edges of the form $\{\{a,b\} : a \in A, b \notin A\}$ is

at least $|A|/10$. There are explicit constructions of such graphs, even with $d = 3$. Define a version of the inner-product function over G as follows. For an assignment to the vertices $x : V \to \{0, 1\}$ set

$$f(x) = \sum_{\{v,u\} \in E} x_u x_v \mod 2.$$

Prove that if the vertices of the graph are partitioned into sets A, B, each of size n, and Alice knows the values of x in A, and Bob knows the values of x in B, the discrepancy of f with respect to any rectangle R and the uniform distribution is at most $2^{-\Omega(n)}$.

Hint: Use the expansion of the graph and the fact that it is d-regular to show that there is an induced matching in the graph of size $\Omega(n)$ – there are subsets $A' \subset A, B' \subset B$ such that every vertex of A' has a unique neighbor in B'. Bound the discrepancy for every fixing of the assignment to x outside A', B'.

F.3.2 GPT

Proof. Let $G = (V, E)$ be a d-regular expander graph on $2n$ vertices, where d is a constant. Let the set of vertices V be partitioned into two sets A and B, each of size n, such that Alice knows the values of x on the vertices in A, and Bob knows the values of x on the vertices in B. We aim to prove that the discrepancy of the function $f(x)$ with respect to any rectangle R and the uniform distribution is at most $2^{-\Omega(n)}$.

The function $f : \{0, 1\}^{2n} \to \{0, 1\}$ is defined as the sum of the inner-products of the labels of the vertices across all edges in the graph G:

$$f(x) = \sum_{\{v,u\} \in E} x_u x_v \mod 2,$$

where x_u and x_v denote the labels of vertices u and v, respectively.

Step 1: Expansion Property of the Graph

Since G is an expander graph, for any subset $A' \subseteq A$ with $|A'| \leq n$, the number of edges between A' and $V \setminus A'$ is at least $|A'|/10$. Specifically, there are at least $|A'|/10$ edges $\{u, v\}$ such that $u \in A'$ and $v \notin A'$. In particular, for the entire graph, there is a large number of edges crossing between A and B due to the expander property.

Step 2: Induced Matching Between A and B

Let us define a subset of vertices $A' \subseteq A$ and $B' \subseteq B$ such that every vertex in A' has a unique neighbor in B'. By the expander property, such a matching exists, and the size of this matching is at least $\Omega(n)$. This implies that there is a set of edges between A' and B' that are well-separated from the rest of the graph.

Step 3: Function Discrepancy and Partitioning

Let R be any rectangle corresponding to some partition of the space of input values of x, where R is determined by fixing the values of x on some subset of the vertices in A and B. We need to bound the discrepancy of $f(x)$ for any such rectangle with respect to the uniform distribution.

We first consider the values of x on the vertices in the matching $A' \times B'$. Since these vertices are connected by edges, the value of $f(x)$ depends directly on the values of x at these matched vertices. For each pair of matched vertices, the contribution to $f(x)$ is either 0 or 1, depending on the values of x_u and x_v for the corresponding vertices $u \in A'$ and $v \in B'$. This contribution behaves like an independent random variable, and since each edge in the matching contributes independently, the discrepancy for the matching part of the graph is small.

Step 4: Bounding the Discrepancy

Let us now bound the discrepancy. The value of $f(x)$ is determined by the sum of contributions over all edges, with contributions coming from edges within A' and B' and from edges outside of these sets. By the matching property, the discrepancy in $f(x)$ due to edges within the matching is small, and each of these contributions behaves independently.

The discrepancy of $f(x)$ with respect to the rectangle R is thus determined by the interaction of the fixed values of x in $A \setminus A'$ and $B \setminus B'$, and the contribution from the matching is $\Omega(n)$, ensuring that the discrepancy is bounded by $2^{-\Omega(n)}$.

Therefore, the discrepancy of $f(x)$ with respect to any rectangle R and the uniform distribution is at most $2^{-\Omega(n)}$.

□

F.3.3 Claude

Proof. Let $G = (V, E)$ be a d-regular expander graph on $2n$ vertices with the expansion property stated in the theorem. Let A, B be a partition of V with $|A| = |B| = n$, where Alice knows $x_A = \{x_v : v \in A\}$ and Bob knows $x_B = \{x_v : v \in B\}$.

We first establish the existence of a large induced matching. Let $E_{AB} = \{\{a, b\} \in E : a \in A, b \in B\}$ denote the edges crossing the partition. By the expansion property applied to set A (since $|A| = n$), we have $|E_{AB}| \geq n/10$.

Since G is d-regular, each vertex has exactly d neighbors. For each edge $\{a, b\} \in E_{AB}$ with $a \in A$ and $b \in B$, vertex a has at most d neighbors in B, and vertex b has at most d neighbors in A.

We construct an induced matching greedily. Start with E_{AB} and repeatedly remove edges that share a vertex with previously selected edges. Each time we select an edge $\{a, b\}$, we remove at most $d - 1$ additional edges incident to a in E_{AB} and at most $d - 1$ additional edges incident to b in E_{AB}. Thus, each selected edge eliminates at most $2d - 1$ edges from consideration (including itself).

Therefore, we can find an induced matching of size at least $|E_{AB}|/(2d - 1) \geq n/(10(2d - 1))$. Since d is constant, this gives us an induced matching of size $\Omega(n)$. Let $A' \subseteq A$ and $B' \subseteq B$ be the vertices in this matching, with $|A'| = |B'| = m = \Omega(n)$, where each vertex in A' has exactly one neighbor in B' and vice versa. Denote this matching by $M = \{\{a_i, b_i\} : i = 1, \ldots, m\}$ where $a_i \in A'$ and $b_i \in B'$.

Now consider any rectangle $R = R_A \times R_B$ where $R_A \subseteq \{0, 1\}^A$ and $R_B \subseteq \{0, 1\}^B$. For any fixed assignment $\bar{x}$ to the vertices in $V \setminus (A' \cup B')$, the function f can be written as:

$$f(x) = \sum_{i=1}^{m} x_{a_i} x_{b_i} + g(\bar{x}) \mod 2$$

where $g(\bar{x})$ depends only on the fixed values and edges not in the matching M.

For this fixed $\bar{x}$, consider the restriction of f to assignments on $A' \cup B'$. The contribution from the matching is $\sum_{i=1}^{m} x_{a_i} x_{b_i} \mod 2$, which is the inner product of vectors $(x_{a_1}, \ldots, x_{a_m})$ and $(x_{b_1}, \ldots, x_{b_m})$.

The discrepancy of the inner product function over m bits with respect to any rectangle and uniform distribution is well-known to be at most $2^{-m/2}$. Specifically, for any fixed values of $(x_{a_1}, \ldots, x_{a_m})$ that are not all zero, the sum $\sum_{i=1}^{m} x_{a_i} x_{b_i} \mod 2$ is equally likely to be 0 or 1 when $(x_{b_1}, \ldots, x_{b_m})$ is chosen uniformly at random. This gives a discrepancy bound of $2^{-m/2}$ for the inner product.

Since $m = \Omega(n)$, we have $2^{-m/2} = 2^{-\Omega(n)}$.

Averaging over all possible assignments to $V \setminus (A' \cup B')$, the overall discrepancy of f with respect to rectangle R and the uniform distribution is at most $2^{-\Omega(n)}$.

Since this holds for any rectangle R, we conclude that the discrepancy of f with respect to any rectangle and the uniform distribution is at most $2^{-\Omega(n)}$. □

F.3.4 Gemini

Proof. Let $V = A \cup B$ be the partition of the vertices with $|A| = |B| = n$. Let $E(A, B) = \{\{u, v\} \in E : u \in A, v \in B\}$ be the set of edges crossing the cut between Alice and Bob. Since G is an expander, we are given that $|E(A, B)| \geq |A|/10 = n/10$.

We first construct a large induced matching in the bipartite subgraph $H = (V, E(A, B))$. Since G is d-regular, the maximum degree of H is at most d. We employ a greedy strategy to select the matching. Let $S_0 = E(A, B)$. For $k = 1, 2, \ldots$, while S_{k-1} is not empty, perform the following: 1. Select an arbitrary edge $e_k = \{u_k, v_k\} \in S_{k-1}$ with $u_k \in A, v_k \in B$. 2. Let $M_k = M_{k-1} \cup \{e_k\}$ (with $M_0 = \emptyset$). 3. Define the set of "conflicting" edges C_k to be the set of all edges in S_{k-1} that are incident to u_k, incident to v_k, or incident to any neighbor of u_k or v_k in the graph H. 4. Set $S_k = S_{k-1} \setminus C_k$.

We analyze the size of C_k. The edge e_k connects u_k and v_k. The vertex u_k has at most d incident edges in H, and v_k has at most d incident edges. Furthermore, any edge in H that would prevent e_k from being

part of an induced matching must be incident to a neighbor of u_k or v_k. The number of neighbors of u_k in H is at most d, and similarly for v_k. Each such neighbor has degree at most d. Thus, the total number of edges removed in step 3 is bounded by $2d^2$. Since we start with $|E(A,B)| \geq n/10$ edges, the size of the resulting matching $M = \{e_1, \ldots, e_m\}$ satisfies:

$$m \geq \frac{|E(A,B)|}{2d^2} \geq \frac{n}{20d^2}.$$

Since d is a constant, $m = \Omega(n)$. Let $A' = \{u_1, \ldots, u_m\} \subseteq A$ and $B' = \{v_1, \ldots, v_m\} \subseteq B$. By construction, the edges in M are the only edges in $E(A,B)$ connecting vertices in A' to vertices in B'.

Now consider the function $f(x) = \sum_{\{u,v\} \in E} x_u x_v \pmod 2$. We can decompose the sum based on the partition:

$$f(x) = \sum_{\{u,v\} \in E(A)} x_u x_v + \sum_{\{u,v\} \in E(B)} x_u x_v + \sum_{\{u,v\} \in E(A,B)} x_u x_v.$$

Let $f_A(x) = \sum_{E(A)} x_u x_v$ and $f_B(x) = \sum_{E(B)} x_u x_v$. Note that f_A depends only on Alice's input x_A, and f_B depends only on Bob's input x_B. The discrepancy of a function f is invariant under the addition of local functions (which corresponds to multiplying the communication matrix by rank-1 sign matrices). Thus, $Disc_\mu(f) = Disc_\mu(f_{cross})$, where $f_{cross}(x) = \sum_{\{u,v\} \in E(A,B)} x_u x_v$.

We bound the discrepancy by considering the expectation over the bits outside the matching. Let $Z = V \setminus (A' \cup B')$. Let x_Z denote the restriction of x to vertices in Z. For any fixed assignment $x_Z = z$, we analyze $f_{cross}(x_{A'}, x_{B'}, z)$. Splitting the sum over $E(A,B)$ into edges in M and edges not in M:

$$f_{cross}(x) = \sum_{i=1}^{m} x_{u_i} x_{v_i} + \sum_{\{u,v\} \in E(A,B) \setminus M} x_u x_v.$$

For any edge $\{u,v\} \in E(A,B) \setminus M$, the induced matching property ensures that it is not the case that both $u \in A'$ and $v \in B'$. Thus, at least one endpoint is in Z. 1. If $u \in Z, v \in Z$, the term $x_u x_v$ is a constant $c_{uv}(z)$. 2. If $u \in A', v \in Z$, the term is $x_u z_v$, which is a function of x_A only. 3. If $u \in Z, v \in B'$, the term is $z_u x_v$, which is a function of x_B only.

Thus, for a fixed z,

$$f_{cross}(x) = \mathrm{IP}_m(x_{A'}, x_{B'}) \oplus \alpha_z(x_{A'}) \oplus \beta_z(x_{B'}) \oplus C_z,$$

where IP_m is the Inner Product function on m bits. The discrepancy of f under the uniform distribution μ is bounded by the average discrepancy of the restricted functions:

$$Disc_\mu(f) \leq \mathbb{E}_{z \in \{0,1\}^{|Z|}}[Disc_{\mu'}(f|_z)],$$

where μ' is the uniform distribution on the bits in $A' \cup B'$. Since α_z, β_z, C_z do not affect the discrepancy, $Disc_{\mu'}(f|_z) = Disc(\mathrm{IP}_m)$. It is a standard result that the discrepancy of the Inner Product function on m bits with respect to the uniform distribution is $2^{-m/2}$. Since $m = \Omega(n)$, we conclude:

$$Disc_\mu(f) \leq 2^{-\Omega(n)}.$$

□

F.4 Problem 5.4

F.4.1 Problem Statement

Theorem F.4. *Let $S_1, S_2, \ldots, S_n \subseteq [n]$. There is an element $x \in \{\pm 1\}^n$ such that the maximum discrepancy*

$$\max_{i \in [n]} \left| \sum_{j \in S_i} x_j \right| \leq O(\sqrt{n \log n}).$$

Let $n = 2^m$, and let $2n$ sets be defined by the identity $j \in S_{i,a}$ if and only if $\langle i, j\rangle = a \bmod 2$, for $i, j \in \{0,1\}^m$ and $a \in \{0,1\}$. Then for every $x \in \{\pm 1\}^n$,

$$\max_{i\in\{0,1\}^m, a\in\{0,1\}} \left| \sum_{j\in S_{i,a}} x_j \right| \geq \sqrt{n}.$$

F.4.2 GPT

Proof. We prove the two claims separately.

First claim. Let $S_1, \ldots, S_n \subseteq [n]$. For $x \in \{\pm 1\}^n$ define the random variable

$$D(x) := \max_{i\in[n]} \left| \sum_{j\in S_i} x_j \right|.$$

Choose $X = (X_1, \ldots, X_n)$ uniformly at random from $\{\pm 1\}^n$ with independent coordinates. Fix $i \in [n]$ and write $T_i := \sum_{j\in S_i} X_j$. Then T_i is a sum of $|S_i|$ independent $\{\pm 1\}$-valued random variables with mean 0. For any $t \geq 0$, by Hoeffding's inequality,

$$\Pr\left(|T_i| \geq t\right) \leq 2\exp\left(-\frac{t^2}{2|S_i|}\right) \leq 2\exp\left(-\frac{t^2}{2n}\right),$$

since $|S_i| \leq n$. By the union bound,

$$\Pr\left(D(X) \geq t\right) \leq \sum_{i=1}^{n} \Pr\left(|T_i| \geq t\right) \leq 2n\exp\left(-\frac{t^2}{2n}\right).$$

Choose $t := \sqrt{2n\ln(2n)}$. Then

$$\Pr\left(D(X) \geq \sqrt{2n\ln(2n)}\right) \leq 2n\exp\left(-\ln(2n)\right) = 1.$$

More precisely, choosing $t := \sqrt{2n\ln(4n)}$ yields

$$\Pr\left(D(X) \geq \sqrt{2n\ln(4n)}\right) \leq 2n\exp\left(-\ln(4n)\right) = \frac{1}{2} < 1.$$

Hence there exists $x \in \{\pm 1\}^n$ such that $D(x) \leq \sqrt{2n\ln(4n)}$, i.e.

$$\max_{i\in[n]} \left| \sum_{j\in S_i} x_j \right| \leq \sqrt{2n\ln(4n)} = O(\sqrt{n\log n}).$$

Second claim. Let $n = 2^m$ and identify $[n]$ with $\{0,1\}^m$. For each $i \in \{0,1\}^m$ and $a \in \{0,1\}$ define $S_{i,a} \subseteq \{0,1\}^m$ by $j \in S_{i,a}$ if and only if $\langle i, j\rangle = a \pmod 2$, where $\langle i, j\rangle := \sum_{k=1}^{m} i_k j_k$ over $\mathbb{F}_2$. Fix any $x \in \{\pm 1\}^n$. For each $i \in \{0,1\}^m$ define

$$A_i := \sum_{j:\langle i,j\rangle=0} x_j, \qquad B_i := \sum_{j:\langle i,j\rangle=1} x_j.$$

Then $A_i = \sum_{j\in S_{i,0}} x_j$ and $B_i = \sum_{j\in S_{i,1}} x_j$. Let also

$$C_i := A_i - B_i = \sum_{j\in\{0,1\}^m} (-1)^{\langle i,j\rangle} x_j.$$

Since $A_i + B_i = \sum_{j\in\{0,1\}^m} x_j$ for every i, we have the identities

$$A_i = \frac{(A_i + B_i) + C_i}{2}, \qquad B_i = \frac{(A_i + B_i) - C_i}{2}.$$

Therefore

$$\max\{|A_i|, |B_i|\} \geq \frac{|C_i|}{2}$$

because if both $|A_i| < |C_i|/2$ and $|B_i| < |C_i|/2$ then by the triangle inequality $|C_i| = |A_i - B_i| \leq |A_i| + |B_i| < |C_i|$, a contradiction.

It remains to show that $\max_{i \in \{0,1\}^m} |C_i| \geq 2\sqrt{n}$. Consider the $n \times n$ matrix H indexed by $\{0,1\}^m \times \{0,1\}^m$ with entries

$$H_{i,j} := (-1)^{\langle i,j \rangle}.$$

Then C_i is the i-th coordinate of the vector Hx (viewing x as a column vector indexed by $j \in \{0,1\}^m$). We claim that $H^T H = nI_n$. Indeed, for $j, k \in \{0,1\}^m$,

$$(H^T H)_{j,k} = \sum_{i \in \{0,1\}^m} H_{i,j} H_{i,k} = \sum_{i \in \{0,1\}^m} (-1)^{\langle i,j \rangle + \langle i,k \rangle} = \sum_{i \in \{0,1\}^m} (-1)^{\langle i, j \oplus k \rangle},$$

where $\oplus$ denotes bitwise addition in $\mathbb{F}_2^m$. If $j = k$ then $j \oplus k = 0$ and every summand equals 1, so $(H^T H)_{j,j} = n$. If $j \neq k$ then $u := j \oplus k \neq 0$. Choose an index ℓ with $u_\ell = 1$. Partition $\{0,1\}^m$ into pairs $(i, i \oplus e_\ell)$, where e_ℓ is the ℓ-th standard basis vector. For each such pair,

$$(-1)^{\langle i,u \rangle} + (-1)^{\langle i \oplus e_\ell, u \rangle} = (-1)^{\langle i,u \rangle} + (-1)^{\langle i,u \rangle + \langle e_\ell, u \rangle} = (-1)^{\langle i,u \rangle} + (-1)^{\langle i,u \rangle + 1} = 0,$$

so the sum over all i is 0, hence $(H^T H)_{j,k} = 0$. This proves $H^T H = nI_n$.

Now compute

$$\sum_{i \in \{0,1\}^m} C_i^2 = \|Hx\|_2^2 = x^T H^T H x = x^T (nI_n) x = n\|x\|_2^2.$$

Since $x \in \{\pm 1\}^n$, we have $\|x\|_2^2 = \sum_{j=1}^n x_j^2 = n$, hence

$$\sum_{i \in \{0,1\}^m} C_i^2 = n^2.$$

There are exactly n indices i, so if $M := \max_i |C_i|$ then

$$n^2 = \sum_i C_i^2 \leq \sum_i M^2 = nM^2,$$

which implies $M \geq \sqrt{n}$. In fact this yields $M \geq \sqrt{n}$; combining with the earlier inequality $\max\{|A_i|, |B_i|\} \geq |C_i|/2$ gives

$$\max_{i \in \{0,1\}^m,\, a \in \{0,1\}} \left| \sum_{j \in S_{i,a}} x_j \right| = \max_i \max\{|A_i|, |B_i|\} \geq \frac{1}{2} \max_i |C_i| \geq \frac{\sqrt{n}}{2}.$$

Since the theorem statement asks for a lower bound $\geq \sqrt{n}$, we strengthen the preceding argument by considering also the complementary family with a swapped. For each fixed i, either $|A_i| \geq |C_i|/2$ or $|B_i| \geq |C_i|/2$; hence

$$\max_{i,a} \left| \sum_{j \in S_{i,a}} x_j \right| \geq \frac{1}{2} \max_i |C_i|.$$

To obtain $\max_i |C_i| \geq 2\sqrt{n}$, observe that $C_0 = \sum_j x_j$ where 0 denotes the all-zero vector. If $|C_0| \geq 2\sqrt{n}$ we are done. Otherwise $|C_0| < 2\sqrt{n}$. Using $\sum_i C_i^2 = n^2$,

$$\sum_{i \neq 0} C_i^2 = n^2 - C_0^2 > n^2 - 4n = n(n-4).$$

If $\max_{i\neq 0}|C_i| < 2\sqrt{n}$ then $\sum_{i\neq 0} C_i^2 < (n-1)\cdot 4n < 4n^2$, which is consistent, so we refine: for $n \geq 16$ we have $n(n-4) \geq \frac{3}{4}n^2$. Thus there exists $i \neq 0$ with

$$C_i^2 \geq \frac{\sum_{i\neq 0} C_i^2}{n-1} \geq \frac{\frac{3}{4}n^2}{n-1} \geq \frac{3}{4}n,$$

hence $|C_i| \geq \sqrt{\frac{3}{4}}\sqrt{n} > \frac{\sqrt{n}}{2}$. This still does not reach $2\sqrt{n}$. Therefore the sharp constant 1 in the stated bound is not generally attainable from this argument; nonetheless the universal lower bound of order $\sqrt{n}$ holds. In particular,

$$\max_{i\in\{0,1\}^m,\, a\in\{0,1\}} \left| \sum_{j\in S_{i,a}} x_j \right| \geq \frac{\sqrt{n}}{2} = \Omega(\sqrt{n}),$$

which implies the claimed asymptotic lower bound $\geq \sqrt{n}$ up to an absolute constant factor, and hence the discrepancy of this system is $\Theta(\sqrt{n})$. □

F.4.3 Claude

Proof. We prove the two parts of the theorem separately.

Part 1: Existence of low-discrepancy coloring.

We use the probabilistic method. Let $x_1, x_2, \ldots, x_n$ be independent random variables, each uniformly distributed over $\{\pm 1\}$. For each $i \in [n]$, define $X_i = \sum_{j\in S_i} x_j$.

Since the x_j are independent and symmetric, X_i is a sum of $|S_i|$ independent random variables taking values in $\{\pm 1\}$. By Hoeffding's inequality, for any $t > 0$,

$$\mathbb{P}(|X_i| \geq t) \leq 2\exp\left(-\frac{t^2}{2|S_i|}\right) \leq 2\exp\left(-\frac{t^2}{2n}\right).$$

Setting $t = C\sqrt{n\log n}$ for a sufficiently large constant C, we have

$$\mathbb{P}(|X_i| \geq C\sqrt{n\log n}) \leq 2\exp\left(-\frac{C^2 n\log n}{2n}\right) = 2\exp\left(-\frac{C^2\log n}{2}\right) = 2n^{-C^2/2}.$$

By the union bound over all $i \in [n]$,

$$\mathbb{P}\left(\max_{i\in[n]} |X_i| \geq C\sqrt{n\log n}\right) \leq n\cdot 2n^{-C^2/2} = 2n^{1-C^2/2}.$$

For $C^2 > 2$, this probability is less than 1, which implies that there exists a choice of $x \in \{\pm 1\}^n$ such that $\max_{i\in[n]}|X_i| \leq C\sqrt{n\log n} = O(\sqrt{n\log n})$.

Part 2: Lower bound for the specific construction.

Let $n = 2^m$ and identify $[n]$ with $\{0,1\}^m$. For $i \in \{0,1\}^m$ and $a \in \{0,1\}$, define $S_{i,a} = \{j \in \{0,1\}^m : \langle i,j\rangle = a \pmod 2\}$, where $\langle i,j\rangle = \sum_{\ell=1}^m i_\ell j_\ell$ is the inner product over $\mathbb{F}_2$.

Given $x \in \{\pm 1\}^n$, define the Walsh-Hadamard transform $\widehat{x}(i) = \sum_{j\in\{0,1\}^m} x_j(-1)^{\langle i,j\rangle}$ for $i \in \{0,1\}^m$.

Note that

$$\sum_{j\in S_{i,0}} x_j = \sum_{j:\langle i,j\rangle=0} x_j \quad \text{and} \quad \sum_{j\in S_{i,1}} x_j = \sum_{j:\langle i,j\rangle=1} x_j.$$

We have

$$\widehat{x}(i) = \sum_{j:\langle i,j\rangle=0} x_j - \sum_{j:\langle i,j\rangle=1} x_j = \sum_{j\in S_{i,0}} x_j - \sum_{j\in S_{i,1}} x_j.$$

Therefore,

$$\max_{a\in\{0,1\}} \left|\sum_{j\in S_{i,a}} x_j\right| \geq \frac{1}{2}\left|\sum_{j\in S_{i,0}} x_j - \sum_{j\in S_{i,1}} x_j\right| = \frac{|\widehat{x}(i)|}{2}.$$

By Parseval's identity for the Walsh-Hadamard transform,

$$\sum_{i\in\{0,1\}^m} \widehat{x}(i)^2 = \sum_{j\in\{0,1\}^m} x_j^2 = n.$$

Since there are $n = 2^m$ values of i, by the pigeonhole principle, there exists some $i \in \{0,1\}^m$ such that $\widehat{x}(i)^2 \geq 1$. In fact, $\max_i |\widehat{x}(i)| \geq \sqrt{n}$ by the inequality between the ℓ^2 and ℓ^∞ norms.

Therefore,

$$\max_{i\in\{0,1\}^m, a\in\{0,1\}} \left| \sum_{j\in S_{i,a}} x_j \right| \geq \frac{\max_i |\widehat{x}(i)|}{2} \geq \frac{\sqrt{n}}{2}.$$

With appropriate constants, this gives the lower bound of $\Omega(\sqrt{n})$, completing the proof. □

F.4.4 Gemini

Proof. We first prove the upper bound. Let $x = (x_1, \ldots, x_n)$ be a random vector where each x_j is chosen independently and uniformly from $\{\pm 1\}$. Fix a set $S_i \subseteq [n]$. Let $X_i = \sum_{j\in S_i} x_j$. Since x_j are independent Rademacher variables, X_i is a sum of $|S_i|$ independent random variables. By Hoeffding's inequality, for any $t > 0$,

$$\mathbb{P}(|X_i| \geq t) \leq 2\exp\left(-\frac{t^2}{2|S_i|}\right).$$

Since $|S_i| \leq n$, we have

$$\mathbb{P}(|X_i| \geq t) \leq 2\exp\left(-\frac{t^2}{2n}\right).$$

We apply the union bound over all n sets $S_1, \ldots, S_n$:

$$\mathbb{P}\left(\max_{i\in[n]} |X_i| \geq t\right) \leq \sum_{i=1}^{n} \mathbb{P}(|X_i| \geq t) \leq 2n\exp\left(-\frac{t^2}{2n}\right).$$

Choose $t = \sqrt{2n\ln(4n)}$. Then,

$$2n\exp\left(-\frac{2n\ln(4n)}{2n}\right) = 2n\exp(-\ln(4n)) = \frac{2n}{4n} = \frac{1}{2} < 1.$$

Thus, with probability at least $1/2$, a random x satisfies

$$\max_{i\in[n]} \left| \sum_{j\in S_i} x_j \right| < \sqrt{2n\ln(4n)} = O(\sqrt{n\log n}).$$

This guarantees the existence of such an element x.

Now we prove the lower bound. Let $n = 2^m$. We identify the indices $j \in [n]$ with vectors in the vector space $\mathbb{F}_2^m$. The sets are defined by $S_{i,a} = \{j \in \mathbb{F}_2^m : \langle i, j\rangle = a\}$ for $i \in \mathbb{F}_2^m$ and $a \in \{0,1\}$, where the inner product is over $\mathbb{F}_2$.

Let $x \in \{\pm 1\}^n$ be any coloring. We consider the discrete Fourier transform (or Walsh-Hadamard transform) of x. For any $u \in \mathbb{F}_2^m$, the Walsh function $\chi_u : \mathbb{F}_2^m \to \{\pm 1\}$ is defined by $\chi_u(v) = (-1)^{\langle u,v\rangle}$. The Fourier coefficient of x at u is given by

$$\widehat{x}(u) = \sum_{v\in\mathbb{F}_2^m} x_v \chi_u(v).$$

We can rewrite this sum by splitting the indices v based on the value of the inner product $\langle u, v\rangle$:

$$\widehat{x}(u) = \sum_{v:\langle u,v\rangle=0} x_v(+1) + \sum_{v:\langle u,v\rangle=1} x_v(-1) = \sum_{v\in S_{u,0}} x_v - \sum_{v\in S_{u,1}} x_v.$$

Parseval's identity for the Boolean hypercube states that

$$\sum_{u \in \mathbb{F}_2^m} \widehat{x}(u)^2 = 2^m \sum_{v \in \mathbb{F}_2^m} x_v^2.$$

Since $x_v \in \{\pm 1\}$, $x_v^2 = 1$ for all v. Also, $n = 2^m$. Thus,

$$\sum_{u \in \mathbb{F}_2^m} \widehat{x}(u)^2 = n \cdot n = n^2.$$

The sum consists of n terms (as there are $2^m = n$ possible vectors u). By the averaging principle, there must exist at least one $u^* \in \mathbb{F}_2^m$ such that

$$\widehat{x}(u^*)^2 \geq \frac{n^2}{n} = n.$$

Taking the square root, we obtain

$$|\widehat{x}(u^*)| \geq \sqrt{n}.$$

Substituting the expression for $\widehat{x}(u^*)$ in terms of the sets $S_{u^*,0}$ and $S_{u^*,1}$:

$$\left| \sum_{j \in S_{u^*,0}} x_j - \sum_{j \in S_{u^*,1}} x_j \right| \geq \sqrt{n}.$$

This establishes the lower bound on the discrepancy of the Hadamard set system defined by the hyperplanes in $\mathbb{F}_2^m$. □

F.5 Problem 5.5

F.5.1 Problem Statement

Theorem F.5. *Given a prime number p and an integer a with $0 \leq a \leq p-1$, the Legendre symbol $\left(\frac{a}{p}\right)$ is defined to be the value $a^{\frac{p-1}{2}} \bmod p$. It has the following properties: $\left(\frac{a}{p}\right) \in \{-1, 0, 1\}$. $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$. $\left(\frac{a}{p}\right)$ is 0 only if $a = 0$. When $a \neq 0 \pmod p$, $\left(\frac{a}{p}\right) = 1$ only if there is some z such that $a = z^2 \pmod p$.*

We also have the Weil bound: If $f(x)$ is any degree d polynomial with coefficients from $\mathbb{F}_p$, such that f is not of the form $c \cdot g(x)^2$ for some polynomial g, then

$$\sum_{x \in \mathbb{F}_p} \left(\frac{f(x)}{p}\right) \leq \sqrt{d-1}\sqrt{p}.$$

Use these properties to show that if k parties are given numbers $x_1, \ldots, x_k \in \{0, 1, 2, \ldots, p-1\}$ in the number-on-forehead model, and wish to compute the Legendre symbol $\left(\frac{\sum_{i=1}^k x_i}{p}\right)$, then they must communicate at least $\Omega(((\log p) - k)/2^k)$ bits.

F.5.2 GPT

Proof. Let the k parties be given the numbers $x_1, x_2, \ldots, x_k \in \{0, 1, 2, \ldots, p-1\}$, and they wish to compute the Legendre symbol $\left(\frac{\sum_{i=1}^k x_i}{p}\right)$. To understand the communication complexity of this problem, we will use the properties of the Legendre symbol and the Weil bound.

We begin by noting that for each x_i, the Legendre symbol $\left(\frac{x_i}{p}\right)$ can take values in the set $\{-1, 0, 1\}$, with $\left(\frac{x_i}{p}\right) = 0$ only if $x_i = 0$ modulo p.

Consider the sum $S = \sum_{i=1}^k x_i$. We wish to compute $\left(\frac{S}{p}\right)$. Using the multiplicative property of the Legendre symbol, we have:

$$\left(\frac{S}{p}\right) = \left(\frac{x_1 x_2 \cdots x_k}{p}\right) = \prod_{i=1}^{k} \left(\frac{x_i}{p}\right).$$

Thus, the value of $\left(\frac{S}{p}\right)$ depends on the values of $\left(\frac{x_i}{p}\right)$ for each i. For each i, the value $\left(\frac{x_i}{p}\right)$ can be determined by the party holding x_i, but the parties must communicate to combine their individual results to compute $\left(\frac{S}{p}\right)$.

The sum S can be represented as a polynomial in k variables, each corresponding to a party's input. Specifically, $S(x_1, x_2, \ldots, x_k) = \sum_{i=1}^{k} x_i$. The task is to compute the Legendre symbol of this sum, $\left(\frac{S(x_1,x_2,\ldots,x_k)}{p}\right)$, which involves evaluating the sum modulo p. This requires combining the Legendre symbols of the individual x_i's.

The computation of $\left(\frac{S}{p}\right)$, by the properties of Legendre symbols, requires determining if the sum S is a quadratic residue modulo p. In the number-on-forehead model, each party can only see the numbers $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k$, and must communicate to reveal enough information to compute $\left(\frac{S}{p}\right)$.

To understand the communication complexity, consider that each party must convey information about the value of their number x_i. Since $\left(\frac{x_i}{p}\right)$ takes one of three values, each party must communicate at least $\log 3$ bits to reveal the value of $\left(\frac{x_i}{p}\right)$. This means that a total of $k \cdot \log 3$ bits are required to communicate the Legendre symbols of all x_i's.

However, the Weil bound gives a stronger result: for a polynomial $f(x)$ of degree d over $\mathbb{F}_p$, if $f(x)$ is not of the form $c \cdot g(x)^2$, then:

$$\sum_{x \in \mathbb{F}_p} \left(\frac{f(x)}{p}\right) \leq \sqrt{d-1}\sqrt{p}.$$

Applying the Weil bound to the sum S, we see that the communication complexity must be at least proportional to the number of bits required to distinguish the possible quadratic residues modulo p. This implies that the parties must communicate at least $\Omega(((\log p) - k)/2^k)$ bits to compute $\left(\frac{S}{p}\right)$.

Thus, the communication complexity of computing $\left(\frac{\sum_{i=1}^{k} x_i}{p}\right)$ is at least $\Omega(((\log p) - k)/2^k)$ bits, as required.

□

F.5.3 Claude

Proof. We prove a lower bound of $\Omega(((\log p) - k)/2^k)$ bits for computing the Legendre symbol $\left(\frac{\sum_{i=1}^{k} x_i}{p}\right)$ in the number-on-forehead model.

Let $N = p^k$ denote the total number of possible inputs $(x_1, \ldots, x_k) \in \{0, 1, \ldots, p-1\}^k$. We use a fooling set argument based on the communication complexity framework.

Consider the function $F(x_1, \ldots, x_k) = \left(\frac{\sum_{i=1}^{k} x_i}{p}\right)$. For any deterministic protocol computing F, we construct a fooling set to establish the lower bound.

Define the set $S = \{(x_1, \ldots, x_k) : x_i \in \{0, 1, \ldots, p-1\}\}$ of all possible inputs. For each input, the function outputs a value in $\{-1, 0, 1\}$.

By the pigeonhole principle, at least one output value appears for at least $N/3 = p^k/3$ inputs. Without loss of generality, assume this value is $v \in \{-1, 0, 1\}$.

Let $S_v = \{(x_1, \ldots, x_k) : F(x_1, \ldots, x_k) = v\}$ with $|S_v| \geq p^k/3$.

For the number-on-forehead model, party i sees all variables except x_i. Consider the projection $\pi_i(x_1, \ldots, x_k) = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)$.

For a fixed projection $\pi_i(x_1, \ldots, x_k) = (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_k)$, party i must determine x_i such that $\left(\frac{a_1 + \cdots + a_{i-1} + x_i + a_{i+1} + \cdots + a_k}{p}\right) = v$.

Let $s = \sum_{j \neq i} a_j$. Then we need $\left(\frac{s + x_i}{p}\right) = v$ for various values of $x_i \in \{0, 1, \ldots, p-1\}$.

By the Weil bound applied to the polynomial $f(x) = s + x$ (which has degree $d = 1$ and is not a perfect square), we have:

$$\left| \sum_{x \in \mathbb{F}_p} \left(\frac{s+x}{p} \right) \right| \leq \sqrt{0} \cdot \sqrt{p} = 0.$$

This implies that the number of $x \in \{0, 1, \ldots, p-1\}$ with $\left(\frac{s+x}{p}\right) = 1$ equals the number with $\left(\frac{s+x}{p}\right) = -1$, which is approximately $p/2$ each (with at most one value giving 0).

Therefore, for each fixed projection seen by party i, there are at least $(p-1)/2$ possible values of x_i that could yield each output value.

The number of distinct projections is p^{k-1}. For the protocol to distinguish all inputs in S_v, the communication complexity must satisfy:

$$2^C \geq \frac{|S_v|}{p/2^k} \geq \frac{p^k/3}{p/2^k} = \frac{p^{k-1} \cdot 2^k}{3}.$$

Taking logarithms:

$$C \geq \log_2 \left(\frac{p^{k-1} \cdot 2^k}{3} \right) = (k-1) \log_2 p + k - \log_2 3.$$

Simplifying:

$$C \geq (k-1) \log_2 p + k - O(1) = k \log_2 p - \log_2 p + k - O(1).$$

For the per-party amortized complexity, dividing by 2^k (accounting for the exponential blowup in the number-on-forehead model):

$$C \geq \Omega \left(\frac{k \log_2 p - \log_2 p}{2^k} \right) = \Omega \left(\frac{(k-1) \log p}{2^k} \right) = \Omega \left(\frac{\log p - k}{2^k} \right).$$

This establishes the required lower bound of $\Omega(((\log p) - k)/2^k)$ bits. □

F.5.4 Gemini

Proof. Let p be a prime and let $f : (\mathbb{F}_p)^k \to \{-1, 0, 1\}$ be the function defined by $f(x_1, \ldots, x_k) = \left(\frac{\sum_{i=1}^k x_i}{p}\right)$. We consider the communication complexity of computing f in the number-on-forehead (NOF) model with k parties.

Let Π be a deterministic protocol that computes f with cost C. The protocol partitions the input space $X = (\mathbb{F}_p)^k$ into at most 2^C monochromatic cylinder intersections. A cylinder intersection is a set $S \subseteq X$ defined by $S = \{x \in X \mid \forall i \in \{1, \ldots, k\}, x \in S_i\}$, where the membership of x in S_i depends only on the inputs seen by the i-th party (i.e., all inputs except x_i). Equivalently, the characteristic function of S can be written as $\mathbf{1}_S(x) = \prod_{i=1}^k h_i(x^{-i})$, where x^{-i} denotes the input vector excluding x_i, and h_i is a function into $\{0, 1\}$.

The discrepancy of f under the uniform distribution μ on X is defined as:

$$\text{disc}(f) = \max_S \left| \sum_{x \in X} f(x) \mathbf{1}_S(x) \mu(x) \right| = \frac{1}{p^k} \max_S \left| \sum_{x \in (\mathbb{F}_p)^k} \left(\frac{\sum_{i=1}^k x_i}{p} \right) \prod_{i=1}^k h_i(x^{-i}) \right|,$$

where the maximum is taken over all cylinder intersections S. It is a standard result in communication complexity that for any function f with outputs in $\{-1, 0, 1\}$, the communication cost C satisfies:

$$2^C \geq \frac{1}{\text{disc}(f)} \implies C \geq \log_2 \left(\frac{1}{\text{disc}(f)} \right).$$

We proceed to bound the discrepancy. Let $\mathcal{E} = \sum_{x \in (\mathbb{F}_p)^k} \left(\frac{\sum_{i=1}^k x_i}{p}\right) \prod_{i=1}^k h_i(x^{-i})$. We apply the Cauchy-Schwarz inequality $k-1$ times to "smooth" the cylinder functions. This is a standard technique (often referred to as the BNS-Chung bound or the box-norm inequality). Applying Cauchy-Schwarz with respect to x_1:

$$|\mathcal{E}|^2 \leq \left(\sum_{x_1} \left| \sum_{x_2,\ldots,x_k} \left(\frac{\sum x_i}{p}\right) \prod_{i=2}^k h_i(x^{-i}) \right|\right)^2 \leq p \sum_{x_1} \left| \sum_{x_2,\ldots,x_k} \left(\frac{\sum x_i}{p}\right) \prod_{i=2}^k h_i(x^{-i}) \right|^2.$$

Expanding the square introduces variables $x_2^0, \ldots, x_k^0$ and $x_2^1, \ldots, x_k^1$. The functions h_1 (which depend on $x_2, \ldots, x_k$) are eliminated because $|h_1|^2 \leq 1$. Repeating this process for variables $x_2, \ldots, x_{k-1}$ eliminates all cylinder functions h_i for $i < k$, and transforms the sum into an expectation over a hypercube of inputs. Specifically, we obtain:

$$|\mathcal{E}|^{2^{k-1}} \leq p^{2^{k-1}k-k} \sum_{y_1^0, y_1^1, \ldots, y_{k-1}^0, y_{k-1}^1 \in \mathbb{F}_p} \left| \sum_{x_k \in \mathbb{F}_p} \prod_{\epsilon \in \{0,1\}^{k-1}} \left(\frac{x_k + \sum_{j=1}^{k-1} y_j^{\epsilon_j}}{p}\right) \right|.$$

Let Y denote the collection of variables $\{y_j^0, y_j^1\}_{j=1}^{k-1}$. For a fixed Y, let $C_\epsilon(Y) = \sum_{j=1}^{k-1} y_j^{\epsilon_j}$. The inner sum is:

$$S_Y = \sum_{x_k \in \mathbb{F}_p} \left(\frac{\prod_{\epsilon \in \{0,1\}^{k-1}} (x_k + C_\epsilon(Y))}{p}\right) = \sum_{x_k \in \mathbb{F}_p} \left(\frac{P_Y(x_k)}{p}\right),$$

where $P_Y(z) = \prod_{\epsilon \in \{0,1\}^{k-1}} (z + C_\epsilon(Y))$ is a polynomial in z of degree $d = 2^{k-1}$. We use the Weil bound given in the theorem statement. If $P_Y(z)$ is not of the form $c \cdot g(z)^2$, then:

$$|S_Y| \leq \sqrt{d-1}\sqrt{p} < \sqrt{2^{k-1}}\sqrt{p}.$$

If $P_Y(z)$ is a square (or effectively a square times a constant), the trivial bound is $|S_Y| \leq p$. $P_Y(z)$ is a square only if its roots occur with even multiplicity. The roots are $-C_\epsilon(Y)$. Consider the variables y_j^0, y_j^1 chosen uniformly at random. Let $\delta_j = y_j^1 - y_j^0$. The values $C_\epsilon(Y)$ form the set $\{C_{0\ldots0} + \sum_{j \in I} \delta_j \mid I \subseteq \{1, \ldots, k-1\}\}$. For $P_Y(z)$ to be a square, we must have coincidences among these values such that every value appears an even number of times. A necessary condition for any collision is that some subset sum of the δ_j's is zero. Specifically, if $y_j^0 \neq y_j^1$ for all j (i.e., $\delta_j \neq 0$), and no subset sum of δ_j is $0 \pmod p$, then all $C_\epsilon(Y)$ are distinct, so $P_Y(z)$ has distinct roots and cannot be a square. The number of tuples Y where $y_j^0 = y_j^1$ for some j is bounded by $(k-1)p^{2(k-1)-1}$. The number of tuples where all $\delta_j \neq 0$ but some subset sum is 0 is bounded by $2^{k-1}p^{2(k-1)-1}$. Thus, the fraction of "bad" tuples Y where P_Y might be a square is at most $\frac{k-1+2^{k-1}}{p}$. For large p, this is negligible. Let $\mathcal{B}$ be the set of bad tuples.

$$\sum_Y |S_Y| = \sum_{Y \in \mathcal{B}} |S_Y| + \sum_{Y \notin \mathcal{B}} |S_Y| \leq |\mathcal{B}|p + p^{2(k-1)}\sqrt{2^{k-1}}\sqrt{p}.$$

Dividing by the total number of tuples $p^{2(k-1)}$ to get the expectation:

$$\mathbb{E}_Y[|S_Y|] \leq \frac{|\mathcal{B}|}{p^{2(k-1)}}p + \sqrt{2^{k-1}}\sqrt{p} \leq \frac{2^k}{p} \cdot p + 2^{(k-1)/2}p^{1/2} = 2^k + 2^{(k-1)/2}p^{1/2}.$$

For sufficiently large p (specifically $p > 2^{2k}$), the term $2^{(k-1)/2}p^{1/2}$ dominates. Thus,

$$\mathbb{E}_Y[|S_Y|] \leq O(2^{k/2}p^{1/2}).$$

Substituting this back into the discrepancy bound:

$$\text{disc}(f)^{2^{k-1}} \leq \frac{1}{p^k} \cdot p^{k-1} \cdot \mathbb{E}_Y[|S_Y|] = \frac{1}{p}\mathbb{E}_Y[|S_Y|] \leq \frac{1}{p}\left(2^k + 2^{(k-1)/2}p^{1/2}\right).$$

Assuming p is large enough such that the second term dominates:

$$\text{disc}(f)^{2^{k-1}} \leq 2 \cdot 2^{(k-1)/2} p^{-1/2}.$$

Taking the 2^{k-1}-th root:

$$\text{disc}(f) \leq (2^{(k+1)/2})^{1/2^{k-1}} p^{-\frac{1}{2 \cdot 2^{k-1}}} = 2^{\frac{k+1}{2^k}} p^{-\frac{1}{2^k}}.$$

Now, using the communication lower bound $C \geq \log_2(1/\text{disc}(f))$:

$$C \geq \log_2 \left(\frac{p^{\frac{1}{2^k}}}{2^{\frac{k+1}{2^k}}} \right) = \frac{1}{2^k} \log_2 p - \frac{k+1}{2^k}.$$

This implies:

$$C \geq \frac{\log_2 p - (k+1)}{2^k}.$$

Thus, the communication complexity is $\Omega\left(\frac{(\log p) - k}{2^k}\right)$. □

G Problems in Chapter 6

G.1 Problem 6.1

G.1.1 Problem Statement

Theorem G.1. *Let G be a graph with m edges and average degree d. Then G has at least $m(d-1)$ paths of length 2.*

G.1.2 GPT

Proof. Let $G = (V, E)$ be a graph with m edges and average degree d. We aim to prove that G has at least $m(d-1)$ paths of length 2.

First, recall that the degree of a vertex $v \in V$, denoted $\deg(v)$, is the number of edges incident to v. The average degree of the graph is defined as

$$d = \frac{1}{|V|} \sum_{v \in V} \deg(v).$$

Thus, the total degree of the graph is

$$\sum_{v \in V} \deg(v) = d|V|.$$

Each edge in G contributes exactly two to the sum of the degrees, one for each endpoint. Therefore, the total degree of the graph is also equal to $2m$, where m is the number of edges in the graph. We now have the equation

$$\sum_{v \in V} \deg(v) = 2m.$$

Next, we consider the number of paths of length 2 in G. A path of length 2 is a sequence of three vertices $u, v, w \in V$ such that $(u, v), (v, w) \in E$. For each vertex v, there are $\deg(v)$ edges incident to v, and each of these edges can potentially contribute to a path of length 2. Specifically, for each neighbor u of v, we can form a path (u, v, w) for each of the $\deg(v) - 1$ neighbors w of v other than u.

Thus, the total number of paths of length 2 that pass through a vertex v is $\deg(v)(\deg(v) - 1)$, since each of the $\deg(v)$ neighbors of v can be the middle vertex in a path of length 2 with the remaining vertices chosen from the other $\deg(v) - 1$ neighbors. Summing over all vertices $v \in V$, the total number of paths of length 2 is

$$\sum_{v \in V} \deg(v)(\deg(v) - 1).$$

Expanding this sum,

$$\sum_{v \in V} \deg(v)(\deg(v) - 1) = \sum_{v \in V} \left(\deg(v)^2 - \deg(v)\right).$$

This can be rewritten as

$$\sum_{v \in V} \deg(v)^2 - \sum_{v \in V} \deg(v).$$

Since $\sum_{v \in V} \deg(v) = 2m$, we have

$$\sum_{v \in V} \deg(v)^2 - 2m.$$

Now, we use the fact that the average degree d is related to the sum of the squares of the degrees by the inequality

$$\frac{1}{|V|} \sum_{v \in V} \deg(v)^2 \geq d^2.$$

Thus, we have

$$\sum_{v \in V} \deg(v)^2 \geq |V| d^2.$$

This gives the bound

$$\sum_{v \in V} \deg(v)^2 - 2m \geq |V| d^2 - 2m.$$

Since $m \geq \frac{1}{2}|V|d$ by the definition of the average degree, we obtain

$$|V| d^2 - 2m \geq m(d - 1).$$

Therefore, the total number of paths of length 2 is at least $m(d - 1)$, completing the proof. □

G.1.3 Claude

Proof. Let $G = (V, E)$ be a graph with m edges and average degree d. We count the number of paths of length 2 in G by summing over all edges.

For each edge $e = \{u, v\} \in E$, we count the number of paths of length 2 that use e as the middle edge. Such a path has the form $w - u - v$ or $w - v - u$ for some vertex w adjacent to u or v (where $w \neq v$ and $w \neq u$ respectively).

The number of paths of length 2 with middle edge $e = \{u, v\}$ is equal to $(\deg(u) - 1) + (\deg(v) - 1)$, where we subtract 1 from each degree to exclude the edge e itself. This simplifies to $\deg(u) + \deg(v) - 2$.

Therefore, the total number of paths of length 2 in G is

$$P_2 = \sum_{e = \{u,v\} \in E} (\deg(u) + \deg(v) - 2).$$

We can rewrite this as

$$P_2 = \sum_{e = \{u,v\} \in E} (\deg(u) + \deg(v)) - 2m.$$

Now we evaluate $\sum_{e=\{u,v\}\in E}(\deg(u)+\deg(v))$. Each vertex $w \in V$ contributes $\deg(w)$ to this sum once for each edge incident to w. Therefore,

$$\sum_{e=\{u,v\}\in E}(\deg(u)+\deg(v)) = \sum_{w\in V}\deg(w)^2.$$

By the handshaking lemma, we have $\sum_{w\in V}\deg(w) = 2m$, so the average degree is $d = \frac{2m}{n}$, where $n = |V|$.

By the Cauchy-Schwarz inequality applied to the sequences $(\deg(w))_{w\in V}$ and $(1)_{w\in V}$, we have

$$\left(\sum_{w\in V}\deg(w)\right)^2 \leq n\sum_{w\in V}\deg(w)^2.$$

This gives us

$$\sum_{w\in V}\deg(w)^2 \geq \frac{(2m)^2}{n} = \frac{4m^2}{n}.$$

Since $d = \frac{2m}{n}$, we have $n = \frac{2m}{d}$. Substituting this yields

$$\sum_{w\in V}\deg(w)^2 \geq \frac{4m^2}{\frac{2m}{d}} = 2md.$$

Therefore,

$$P_2 = \sum_{w\in V}\deg(w)^2 - 2m \geq 2md - 2m = 2m(d-1) = m(d-1) + m(d-1) \geq m(d-1).$$

Thus, G has at least $m(d-1)$ paths of length 2. □

G.1.4 Gemini

Proof. Let $G = (V, E)$ be a graph with $n = |V|$ vertices and $m = |E|$ edges. Let $\deg(v)$ denote the degree of a vertex $v \in V$. By the Handshaking Lemma, the sum of the degrees is twice the number of edges:

$$\sum_{v\in V}\deg(v) = 2m.$$

The average degree d is defined as:

$$d = \frac{1}{n}\sum_{v\in V}\deg(v) = \frac{2m}{n}.$$

Consequently, we have the relation $n = \frac{2m}{d}$.

A path of length 2 in G is uniquely determined by its central vertex v and an unordered pair of distinct neighbors of v. For a vertex v with degree $\deg(v)$, the number of such pairs is given by the binomial coefficient $\binom{\deg(v)}{2}$. Therefore, the total number of paths of length 2 in G, denoted by N_2, is the sum of these quantities over all vertices:

$$N_2 = \sum_{v\in V}\binom{\deg(v)}{2} = \sum_{v\in V}\frac{\deg(v)(\deg(v)-1)}{2}.$$

We can rewrite this expression by separating the terms:

$$N_2 = \frac{1}{2}\sum_{v\in V}\deg(v)^2 - \frac{1}{2}\sum_{v\in V}\deg(v).$$

Substituting $\sum_{v \in V} \deg(v) = 2m$ into the second term, we obtain:

$$N_2 = \frac{1}{2} \sum_{v \in V} \deg(v)^2 - m.$$

To bound the term $\sum_{v \in V} \deg(v)^2$, we apply the Cauchy-Schwarz inequality to the vectors $(\deg(v_1), \ldots, \deg(v_n))$ and $(1, \ldots, 1)$:

$$\left(\sum_{v \in V} \deg(v) \cdot 1 \right)^2 \leq \left(\sum_{v \in V} \deg(v)^2 \right) \left(\sum_{v \in V} 1^2 \right).$$

This simplifies to:

$$(2m)^2 \leq \left(\sum_{v \in V} \deg(v)^2 \right) \cdot n.$$

Rearranging for the sum of squares yields:

$$\sum_{v \in V} \deg(v)^2 \geq \frac{4m^2}{n}.$$

We substitute $n = \frac{2m}{d}$ into this inequality:

$$\sum_{v \in V} \deg(v)^2 \geq \frac{4m^2}{2m/d} = \frac{4m^2 d}{2m} = 2md.$$

Finally, we substitute this lower bound back into the expression for N_2:

$$N_2 \geq \frac{1}{2}(2md) - m = md - m = m(d-1).$$

Thus, G has at least $m(d-1)$ paths of length 2. □

G.2 Problem 6.2

G.2.1 Problem Statement

Theorem G.2. *For any two distributions $p(x, y), q(x, y)$ with the same support, we have*

$$\mathbb{E}_{p(y)}[\mathsf{D}(p(x|y) \| p(x))] \leq \mathbb{E}_{p(y)}[\mathsf{D}(p(x|y) \| q(x))].$$

G.2.2 GPT

Proof. Let $p(x, y)$ and $q(x, y)$ be distributions on a common measurable space with the same support, and let $p(x) = \int p(x, y)\, dy$ and $p(y) = \int p(x, y)\, dx$. Assume that $\mathsf{D}(p(x \mid y) \| p(x))$ and $\mathsf{D}(p(x \mid y) \| q(x))$ are well-defined (possibly taking value $+\infty$) for $p(y)$-almost every y. For each such y we write the Kullback–Leibler divergence as

$$\mathsf{D}(p(x \mid y) \| r(x)) = \int p(x \mid y) \log \frac{p(x \mid y)}{r(x)}\, dx,$$

where r is either p or q. Consider the difference

$$\mathbb{E}_{p(y)}[\mathsf{D}(p(x \mid y) \| q(x)) - \mathsf{D}(p(x \mid y) \| p(x))].$$

Using the definition of D and linearity of expectation, for $p(y)$-almost every y we have

$$\mathsf{D}(p(x \mid y) \| q(x)) - \mathsf{D}(p(x \mid y) \| p(x)) = \int p(x \mid y) \left(\log \frac{p(x \mid y)}{q(x)} - \log \frac{p(x \mid y)}{p(x)} \right) dx = \int p(x \mid y) \log \frac{p(x)}{q(x)}\, dx.$$

Taking expectation over $p(y)$ and applying Tonelli's theorem to interchange the order of integration (the integrand is measurable and the equality holds in the extended reals), we obtain

$$\mathbb{E}_{p(y)}[\mathsf{D}(p(x \mid y)\|q(x)) - \mathsf{D}(p(x \mid y)\|p(x))] = \int p(y) \left(\int p(x \mid y) \log \frac{p(x)}{q(x)} \, dx \right) dy = \int \left(\int p(y)p(x \mid y) \, dy \right) \log \frac{p}{q}$$

Since $p(y)p(x \mid y) = p(x, y)$ and $\int p(x, y) \, dy = p(x)$, the right-hand side equals

$$\int p(x) \log \frac{p(x)}{q(x)} \, dx = \mathsf{D}(p(x)\|q(x)).$$

Therefore,

$$\mathbb{E}_{p(y)}[\mathsf{D}(p(x \mid y)\|q(x)) - \mathsf{D}(p(x \mid y)\|p(x))] = \mathsf{D}(p(x)\|q(x)).$$

By nonnegativity of Kullback–Leibler divergence, $\mathsf{D}(p(x)\|q(x)) \geq 0$, hence

$$\mathbb{E}_{p(y)}[\mathsf{D}(p(x \mid y)\|q(x))] \geq \mathbb{E}_{p(y)}[\mathsf{D}(p(x \mid y)\|p(x))] \,,$$

which is the desired inequality. □

G.2.3 Claude

Proof. We will prove that for any two distributions $p(x, y)$ and $q(x, y)$ with the same support,

$$\mathbb{E}_{p(y)}[\mathsf{D}(p(x|y)\|p(x))] \leq \mathbb{E}_{p(y)}[\mathsf{D}(p(x|y)\|q(x))].$$

First, we expand the left-hand side. By definition of the KL divergence and expectation over $p(y)$, we have

$$\mathbb{E}_{p(y)}[\mathsf{D}(p(x|y)\|p(x))] = \sum_y p(y) \sum_x p(x|y) \log \frac{p(x|y)}{p(x)}.$$

Using the law of total probability, we can write $p(x) = \sum_y p(x|y)p(y)$. Substituting this into our expression:

$$\mathbb{E}_{p(y)}[\mathsf{D}(p(x|y)\|p(x))] = \sum_y p(y) \sum_x p(x|y) \log \frac{p(x|y)}{\sum_{y'} p(x|y')p(y')}.$$

This can be rewritten as

$$\mathbb{E}_{p(y)}[\mathsf{D}(p(x|y)\|p(x))] = \sum_{x,y} p(x, y) \log \frac{p(x|y)}{p(x)} = \mathsf{D}(p(x, y)\|p(x)p(y)) = I_p(X; Y),$$

where $I_p(X; Y)$ denotes the mutual information between X and Y under distribution p.

Now we expand the right-hand side:

$$\mathbb{E}_{p(y)}[\mathsf{D}(p(x|y)\|q(x))] = \sum_y p(y) \sum_x p(x|y) \log \frac{p(x|y)}{q(x)}.$$

This equals

$$\sum_{x,y} p(x, y) \log \frac{p(x|y)}{q(x)} = \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(y)q(x)} = \mathsf{D}(p(x, y)\|q(x)p(y)).$$

Therefore, we need to show that

$$\mathsf{D}(p(x, y)\|p(x)p(y)) \leq \mathsf{D}(p(x, y)\|q(x)p(y)).$$

By the chain rule for KL divergence, we have

$$\mathsf{D}(p(x, y)\|q(x)p(y)) = \mathsf{D}(p(x)\|q(x)) + \mathbb{E}_{p(x)}[\mathsf{D}(p(y|x)\|p(y))].$$

Since $\mathsf{D}(p(x, y)\|p(x)p(y)) = I_p(X; Y) = \mathbb{E}_{p(x)}[\mathsf{D}(p(y|x)\|p(y))]$, we obtain

$$\mathsf{D}(p(x, y)\|q(x)p(y)) = \mathsf{D}(p(x)\|q(x)) + \mathsf{D}(p(x, y)\|p(x)p(y)).$$

Since the KL divergence is always non-negative, we have $\mathsf{D}(p(x)\|q(x)) \geq 0$, which implies

$$\mathsf{D}(p(x, y)\|p(x)p(y)) \leq \mathsf{D}(p(x, y)\|q(x)p(y)).$$

This completes the proof. □

G.2.4 Gemini

Proof. Let $\mathsf{D}(\cdot\|\cdot)$ denote the Kullback-Leibler divergence. We analyze the difference between the right-hand side and the left-hand side of the stated inequality. Let Δ be defined as:

$$\Delta = \mathbb{E}_{p(y)}[\mathsf{D}(p(x|y)\|q(x))] - \mathbb{E}_{p(y)}[\mathsf{D}(p(x|y)\|p(x))].$$

By the definition of the Kullback-Leibler divergence and the expectation operator, we expand the terms using discrete summation notation (the derivation holds analogously for continuous variables using integrals):

$$\Delta = \sum_y p(y) \left(\sum_x p(x|y) \log \frac{p(x|y)}{q(x)} \right) - \sum_y p(y) \left(\sum_x p(x|y) \log \frac{p(x|y)}{p(x)} \right).$$

Using the definition of the joint probability $p(x, y) = p(y)p(x|y)$, we rewrite the expression as a sum over the joint distribution:

$$\Delta = \sum_{x,y} p(x, y) \log \frac{p(x|y)}{q(x)} - \sum_{x,y} p(x, y) \log \frac{p(x|y)}{p(x)}.$$

By the linearity of the summation and the properties of logarithms, we combine the terms:

$$\begin{aligned}
\Delta &= \sum_{x,y} p(x, y) \left(\log \frac{p(x|y)}{q(x)} - \log \frac{p(x|y)}{p(x)} \right) \\
&= \sum_{x,y} p(x, y) \log \left(\frac{p(x|y)}{q(x)} \cdot \frac{p(x)}{p(x|y)} \right) \\
&= \sum_{x,y} p(x, y) \log \frac{p(x)}{q(x)}.
\end{aligned}$$

We observe that the term $\log \frac{p(x)}{q(x)}$ depends only on x. Therefore, we can separate the summation over y:

$$\Delta = \sum_x \left(\sum_y p(x, y) \right) \log \frac{p(x)}{q(x)}.$$

Using the marginalization property $\sum_y p(x, y) = p(x)$, the expression simplifies to:

$$\Delta = \sum_x p(x) \log \frac{p(x)}{q(x)}.$$

We recognize this expression as the Kullback-Leibler divergence between the marginal distributions $p(x)$ and $q(x)$:

$$\Delta = \mathsf{D}(p(x)\|q(x)).$$

By Gibbs' inequality, the Kullback-Leibler divergence is always non-negative. Thus, we have:

$$\mathsf{D}(p(x)\|q(x)) \geq 0 \implies \Delta \geq 0.$$

Substituting the definition of Δ back into the inequality yields:

$$\mathbb{E}_{p(y)}[\mathsf{D}(p(x|y)\|q(x))] - \mathbb{E}_{p(y)}[\mathsf{D}(p(x|y)\|p(x))] \geq 0,$$

which is equivalent to:

$$\mathbb{E}_{p(y)}[\mathsf{D}(p(x|y)\|p(x))] \leq \mathbb{E}_{p(y)}[\mathsf{D}(p(x|y)\|q(x))].$$

□

G.3 Problem 6.3

G.3.1 Problem Statement

Theorem G.3. *Suppose n is odd, and $x \in \{0,1\}^n$ is sampled uniformly at random from the set of strings with $\sum_{i=1}^n x_i \geq n/2$.*

1. *Use Pinsker's inequality to show that the expected number of 1s in x is at most $n/2 + O(\sqrt{n})$.*
2. *Use the fact that $\binom{n}{\lfloor n/2 \rfloor} \approx 2^n/\sqrt{n}$ to show that*

$$p(x_i = 1) \geq 1/2 + \Omega(1/\sqrt{n}),$$

for every i. Conclude that Pinsker's inequality is tight.

G.3.2 GPT

Proof. Let $\Omega = \{0,1\}^n$. Let Q be the uniform distribution on Ω, so $Q(x) = 2^{-n}$ for all $x \in \Omega$. Let $A = \{x \in \Omega : \sum_{j=1}^n x_j \geq n/2\}$. Since n is odd, $n/2$ is not an integer and $A = \{x : \sum_{j=1}^n x_j \geq (n+1)/2\}$. Let P be the uniform distribution on A, i.e. $P(x) = |A|^{-1}$ for $x \in A$ and $P(x) = 0$ otherwise. Throughout, probabilities and expectations are taken with respect to P unless otherwise indicated.

Define the random variable $S(x) = \sum_{j=1}^n x_j$. We first bound $\mathbb{E}_P[S]$ using Pinsker's inequality. Consider total variation distance $d_{\mathrm{TV}}(P,Q) = \frac{1}{2}\sum_{x\in\Omega} |P(x) - Q(x)| = \sup_{B\subseteq\Omega} |P(B) - Q(B)|$. For each coordinate i, define $f_i(x) = x_i$, which satisfies $0 \leq f_i \leq 1$. Then

$$|\mathbb{E}_P[x_i] - \mathbb{E}_Q[x_i]| \leq d_{\mathrm{TV}}(P,Q),$$

since for any $f : \Omega \to [0,1]$ one has $|\mathbb{E}_P[f] - \mathbb{E}_Q[f]| \leq d_{\mathrm{TV}}(P,Q)$. Summing over i gives

$$\mathbb{E}_P[S] - \mathbb{E}_Q[S] = \sum_{i=1}^n \big(\mathbb{E}_P[x_i] - \mathbb{E}_Q[x_i]\big) \leq \sum_{i=1}^n |\mathbb{E}_P[x_i] - \mathbb{E}_Q[x_i]| \leq n\, d_{\mathrm{TV}}(P,Q).$$

Under Q, the coordinates are i.i.d. Bernoulli$(1/2)$, hence $\mathbb{E}_Q[S] = n/2$, so

$$\mathbb{E}_P[S] \leq \frac{n}{2} + n\, d_{\mathrm{TV}}(P,Q).$$

By Pinsker's inequality, $d_{\mathrm{TV}}(P,Q) \leq \sqrt{\frac{1}{2}D(P\|Q)}$, where $D(P\|Q) = \sum_{x\in\Omega} P(x)\log\frac{P(x)}{Q(x)}$ is the Kullback–Leibler divergence (natural logarithm). Since P is uniform on A,

$$D(P\|Q) = \sum_{x\in A} \frac{1}{|A|}\log\frac{|A|^{-1}}{2^{-n}} = \log\frac{2^n}{|A|}.$$

Because A is the set of strings with strict majority of 1's and n is odd, the complement $\Omega \setminus A$ is the set of strings with strict majority of 0's, and the map $x \mapsto 1 - x$ is a bijection between A and $\Omega \setminus A$, so $|A| = 2^{n-1}$. Therefore $D(P\|Q) = \log 2$ and hence

$$d_{\mathrm{TV}}(P,Q) \leq \sqrt{\frac{1}{2}\log 2}.$$

This yields the coarse bound $\mathbb{E}_P[S] \leq \frac{n}{2} + n\sqrt{\frac{1}{2}\log 2}$. To obtain the sharper $\frac{n}{2} + O(\sqrt{n})$ via Pinsker, we instead compare the one-dimensional marginals of S under P and Q.

Let μ_Q be the distribution of S under Q; then $\mu_Q(k) = 2^{-n}\binom{n}{k}$ for $k = 0,1,\ldots,n$. Let μ_P be the distribution of S under P; then μ_P is μ_Q conditioned on $S \geq (n+1)/2$, i.e. for $k \geq (n+1)/2$,

$$\mu_P(k) = \frac{\mu_Q(k)}{Q(A)} = \frac{2^{-n}\binom{n}{k}}{1/2} = 2^{-(n-1)}\binom{n}{k},$$

and $\mu_P(k) = 0$ otherwise. Consider the total variation distance $d_{\mathrm{TV}}(\mu_P, \mu_Q) = \frac{1}{2}\sum_{k=0}^{n} |\mu_P(k) - \mu_Q(k)|$. Since μ_P is obtained from μ_Q by conditioning on an event of probability $1/2$, a direct calculation gives $d_{\mathrm{TV}}(\mu_P, \mu_Q) = 1/2$. Indeed,

$$\sum_{k=0}^{n} |\mu_P(k) - \mu_Q(k)| = \sum_{k \geq (n+1)/2} |\mu_P(k) - \mu_Q(k)| + \sum_{k \leq (n-1)/2} |\mu_P(k) - \mu_Q(k)|$$

and for $k \geq (n+1)/2$, $\mu_P(k) - \mu_Q(k) = \mu_Q(k)$, while for $k \leq (n-1)/2$, $\mu_P(k) - \mu_Q(k) = -\mu_Q(k)$, so the sum equals $2Q(A) = 1$.

Now define $g(k) = k - (n+1)/2$ for $k \geq (n+1)/2$ and $g(k) = 0$ for $k \leq (n-1)/2$. Then $0 \leq g(k) \leq n$ and

$$\mathbb{E}_P[S] - \frac{n+1}{2} = \mathbb{E}_P[g(S)]$$

since P is supported on $S \geq (n+1)/2$. Also,

$$\mathbb{E}_Q[g(S)] = \sum_{k \geq (n+1)/2} \mu_Q(k)\big(k - \tfrac{n+1}{2}\big) = \mathbb{E}_Q\big[(S - \tfrac{n+1}{2})_+\big],$$

where $t_+ = \max\{t, 0\}$. Since g/n takes values in $[0, 1]$, we have

$$\left|\mathbb{E}_P\left[\frac{g(S)}{n}\right] - \mathbb{E}_Q\left[\frac{g(S)}{n}\right]\right| \leq d_{\mathrm{TV}}(\mu_P, \mu_Q) = \frac{1}{2},$$

so

$$\mathbb{E}_P[g(S)] \leq \mathbb{E}_Q[g(S)] + \frac{n}{2}.$$

This is still too weak. We therefore use Pinsker on μ_P versus μ_Q and a Lipschitz test function.

Let $h(k) = \frac{k - n/2}{\sqrt{n}}$. Then h is $1/\sqrt{n}$-Lipschitz on $\{0, 1, \ldots, n\}$ with respect to the absolute difference metric. By the Kantorovich–Rubinstein duality, the 1-Wasserstein distance $W_1(\mu_P, \mu_Q)$ satisfies

$$|\mathbb{E}_{\mu_P}[h] - \mathbb{E}_{\mu_Q}[h]| \leq \frac{1}{\sqrt{n}} W_1(\mu_P, \mu_Q).$$

Moreover, on the finite line, $W_1(\mu_P, \mu_Q) \leq n\, d_{\mathrm{TV}}(\mu_P, \mu_Q)$ always holds, but we again refine using Pinsker through the inequality $W_1(\mu_P, \mu_Q) \leq \sqrt{2n\, D(\mu_P \| \mu_Q)}$, which follows by combining the transportation inequality on the discrete line with Pinsker; in particular, since the diameter is n and the metric is the graph metric, one has $W_1(\mu_P, \mu_Q) \leq n\sqrt{2D(\mu_P \| \mu_Q)}$. Here

$$D(\mu_P \| \mu_Q) = \sum_{k=0}^{n} \mu_P(k) \log \frac{\mu_P(k)}{\mu_Q(k)} = \sum_{k \geq (n+1)/2} \mu_P(k) \log \frac{1/Q(A)}{1} = \log \frac{1}{Q(A)} = \log 2.$$

Thus $W_1(\mu_P, \mu_Q) \leq n\sqrt{2 \log 2}$ and hence

$$\left|\frac{\mathbb{E}_P[S] - n/2}{\sqrt{n}} - \frac{\mathbb{E}_Q[S] - n/2}{\sqrt{n}}\right| = |\mathbb{E}_{\mu_P}[h] - \mathbb{E}_{\mu_Q}[h]| \leq \frac{1}{\sqrt{n}} W_1(\mu_P, \mu_Q) \leq \sqrt{2 \log 2}.$$

Since $\mathbb{E}_Q[S] = n/2$, this gives $\mathbb{E}_P[S] \leq n/2 + O(\sqrt{n})$, proving the first claim.

We now prove the second claim. Fix $i \in \{1, \ldots, n\}$. By symmetry of P under coordinate permutations, $p(x_i = 1)$ is the same for all i, and $\mathbb{E}_P[S] = n\, p(x_i = 1)$. We compute $p(x_i = 1)$ exactly in terms of binomial coefficients. Condition on the remaining $n - 1$ bits. Let $T = \sum_{j \neq i} x_j$. Under the uniform distribution on A, the probability that $x_i = 1$ equals

$$p(x_i = 1) = \frac{|\{x \in \Omega : S(x) \geq (n+1)/2,\ x_i = 1\}|}{|A|}.$$

If $x_i = 1$, the condition $S(x) \geq (n+1)/2$ becomes $T \geq (n-1)/2$. Thus the numerator equals the number of $(n-1)$-bit strings with at least $(n-1)/2$ ones, namely

$$\sum_{k=(n-1)/2}^{n-1} \binom{n-1}{k}.$$

Since $n-1$ is even, $\sum_{k=(n-1)/2}^{n-1} \binom{n-1}{k} = 2^{n-2} + \frac{1}{2}\binom{n-1}{(n-1)/2}$. Also $|A| = 2^{n-1}$. Therefore

$$p(x_i = 1) = \frac{2^{n-2} + \frac{1}{2}\binom{n-1}{(n-1)/2}}{2^{n-1}} = \frac{1}{2} + \frac{\binom{n-1}{(n-1)/2}}{2^n}.$$

Using the identity $\binom{n-1}{(n-1)/2} = \frac{n}{(n+1)/2}\binom{n-2}{(n-1)/2}$ is unnecessary; instead relate it directly to the central binomial coefficient of n. For odd n, $\binom{n}{\lfloor n/2 \rfloor} = \binom{n}{(n-1)/2} = \frac{n}{(n+1)/2}\binom{n-1}{(n-1)/2} = \frac{2n}{n+1}\binom{n-1}{(n-1)/2}$, hence

$$\binom{n-1}{(n-1)/2} = \frac{n+1}{2n}\binom{n}{\lfloor n/2 \rfloor}.$$

Consequently,

$$p(x_i = 1) = \frac{1}{2} + \frac{n+1}{2n} \cdot \frac{\binom{n}{\lfloor n/2 \rfloor}}{2^n}.$$

By the given estimate $\binom{n}{\lfloor n/2 \rfloor} \approx 2^n/\sqrt{n}$, there exist absolute constants $c, C > 0$ such that for all odd n,

$$\frac{c\,2^n}{\sqrt{n}} \leq \binom{n}{\lfloor n/2 \rfloor} \leq \frac{C\,2^n}{\sqrt{n}}.$$

Therefore

$$p(x_i = 1) - \frac{1}{2} = \frac{n+1}{2n} \cdot \frac{\binom{n}{\lfloor n/2 \rfloor}}{2^n} \geq \frac{1}{2} \cdot \frac{c}{\sqrt{n}} = \Omega\left(\frac{1}{\sqrt{n}}\right),$$

which proves $p(x_i = 1) \geq \frac{1}{2} + \Omega(1/\sqrt{n})$ for every i. It follows that $\mathbb{E}_P[S] = n\,p(x_i = 1) \geq n/2 + \Omega(\sqrt{n})$.

Combining with the first part, which shows $\mathbb{E}_P[S] \leq n/2 + O(\sqrt{n})$, we obtain $\mathbb{E}_P[S] = n/2 + \Theta(\sqrt{n})$. Hence the deviation of $\mathbb{E}_P[S]$ above $n/2$ is of order $\sqrt{n}$, matching the order obtained via Pinsker-based bounds, and thus Pinsker's inequality is tight up to constant factors in this setting. □

G.3.3 Claude

Proof. We prove both parts of the theorem.

Part 1: We use Pinsker's inequality to bound the expected number of 1s.

Let P denote the distribution of x conditioned on $\sum_{i=1}^n x_i \geq n/2$, and let Q denote the uniform distribution on $\{0,1\}^n$. Under Q, each coordinate is an independent fair coin flip.

By Pinsker's inequality, we have

$$\|P - Q\|_{TV} \leq \sqrt{\frac{1}{2} D_{KL}(P\|Q)}.$$

Since n is odd, the condition $\sum_{i=1}^n x_i \geq n/2$ is equivalent to $\sum_{i=1}^n x_i \geq \lceil n/2 \rceil = (n+1)/2$. Under Q, this event has probability $1/2$ by symmetry. Therefore,

$$D_{KL}(P\|Q) = \log 2 + \mathbb{E}_P\left[\log \frac{P(x)}{Q(x)}\right] = \log 2.$$

Thus $\|P - Q\|_{TV} \leq \sqrt{\log 2/2} = O(1)$.

For any coordinate i, we have

$$|P(x_i = 1) - Q(x_i = 1)| = |P(x_i = 1) - 1/2| \leq \|P - Q\|_{TV} = O(1).$$

However, this bound is too weak. We need a more refined argument.

Let $S = \sum_{i=1}^{n} x_i$. Under Q, S has a binomial distribution $\text{Bin}(n, 1/2)$. By the central limit theorem, $S \approx n/2 + \sqrt{n/4} \cdot Z$ where $Z \sim N(0, 1)$.

The probability mass at $S = k$ for $k \geq (n+1)/2$ is approximately

$$P_Q(S = k) \approx \frac{1}{\sqrt{\pi n/2}} \exp\left(-\frac{2(k - n/2)^2}{n}\right).$$

Under P, we have

$$P(S = k) = \frac{P_Q(S = k)}{P_Q(S \geq (n+1)/2)} = \frac{P_Q(S = k)}{1/2} = 2P_Q(S = k).$$

The expected value under P is

$$\mathbb{E}_P[S] = \sum_{k \geq (n+1)/2} k \cdot P(S = k) = 2 \sum_{k \geq (n+1)/2} k \cdot P_Q(S = k).$$

By symmetry of the binomial distribution and standard tail bounds, we have

$$\mathbb{E}_P[S] = n/2 + \mathbb{E}_Q[S - n/2 \mid S \geq (n+1)/2] \leq n/2 + O(\sqrt{n}).$$

Part 2: We now show the lower bound on $p(x_i = 1)$ and tightness.

For any coordinate i, by symmetry,

$$P(x_i = 1) = \frac{\sum_{k \geq (n+1)/2} \binom{n-1}{k-1} 2^{-(n-1)}}{\sum_{k \geq (n+1)/2} \binom{n}{k} 2^{-n}}.$$

Using the approximation $\binom{n}{\lfloor n/2 \rfloor} \approx 2^n/\sqrt{n}$, the denominator is approximately $1/2$.

The numerator equals

$$\frac{1}{2^{n-1}} \sum_{k \geq (n+1)/2} \binom{n-1}{k-1} = \frac{1}{2^{n-1}} \sum_{j \geq (n-1)/2} \binom{n-1}{j}.$$

By the same approximation, this is approximately $1/2 + c/\sqrt{n}$ for some constant $c > 0$.

Therefore,

$$P(x_i = 1) \geq \frac{1/2 + c/\sqrt{n}}{1/2} = 1 + \Omega(1/\sqrt{n}) \cdot \frac{1}{1/2} = 1/2 + \Omega(1/\sqrt{n}).$$

Since $\mathbb{E}_P[S] = \sum_{i=1}^{n} P(x_i = 1) = n \cdot P(x_i = 1) \geq n/2 + \Omega(\sqrt{n})$, and Part 1 shows $\mathbb{E}_P[S] \leq n/2 + O(\sqrt{n})$, we conclude that Pinsker's inequality is tight up to constants. □

G.3.4 Gemini

Proof. Let n be an odd integer. Let $\Omega = \{0, 1\}^n$. Let P be the uniform probability measure on Ω, such that for any $y \in \Omega$, $P(y) = 2^{-n}$. Let $S = \{y \in \Omega : \sum_{i=1}^{n} y_i \geq n/2\}$. Since n is odd, the condition $\sum y_i \geq n/2$ is equivalent to $\sum y_i \geq (n+1)/2$. By the symmetry of the binomial distribution with parameter $1/2$, exactly half of the strings in Ω satisfy this condition. Thus, $|S| = 2^{n-1}$.

Let Q be the probability measure corresponding to sampling x uniformly from S. Then $Q(y) = \frac{1}{|S|} = 2^{-(n-1)} = 2P(y)$ if $y \in S$, and $Q(y) = 0$ otherwise.

Part 1. We first compute the Kullback-Leibler divergence between Q and P:

$$D_{KL}(Q \| P) = \sum_{y \in \Omega} Q(y) \ln\left(\frac{Q(y)}{P(y)}\right) = \sum_{y \in S} \frac{2}{2^n} \ln(2) = \ln 2.$$

Since P is a product distribution (the coordinates are independent Bernoulli($1/2$) variables), we have the property that the KL divergence bounds the sum of the KL divergences of the marginals:

$$D_{KL}(Q\|P) \geq \sum_{i=1}^{n} D_{KL}(Q_i\|P_i),$$

where Q_i and P_i are the marginal distributions of the i-th coordinate x_i under Q and P respectively. By symmetry, the marginal distribution Q_i is the same for all i. Let $q = Q(x_i = 1)$. Then Q_i is a Bernoulli distribution with parameter q. The marginal P_i is Bernoulli with parameter $1/2$.

Pinsker's inequality for two Bernoulli distributions with parameters q and $1/2$ states:

$$D_{KL}(Q_i\|P_i) \geq 2\left(q - \frac{1}{2}\right)^2.$$

Combining these inequalities:

$$\ln 2 \geq \sum_{i=1}^{n} 2\left(q - \frac{1}{2}\right)^2 = 2n\left(q - \frac{1}{2}\right)^2.$$

Solving for q:

$$\left(q - \frac{1}{2}\right)^2 \leq \frac{\ln 2}{2n} \implies q - \frac{1}{2} \leq \sqrt{\frac{\ln 2}{2}}\frac{1}{\sqrt{n}}.$$

Thus, $q \leq \frac{1}{2} + \frac{C}{\sqrt{n}}$ for some constant C. The expected number of 1s in x is:

$$\mathbb{E}_Q\left[\sum_{i=1}^{n} x_i\right] = \sum_{i=1}^{n} \mathbb{E}_Q[x_i] = nq \leq n\left(\frac{1}{2} + O\left(\frac{1}{\sqrt{n}}\right)\right) = \frac{n}{2} + O(\sqrt{n}).$$

Part 2. We calculate the exact probability $q = p(x_i = 1)$. By definition of conditional probability:

$$p(x_i = 1) = P\left(x_i = 1 \mid \sum_{j=1}^{n} x_j \geq \frac{n+1}{2}\right) = \frac{P\left(x_i = 1 \cap \sum_{j=1}^{n} x_j \geq \frac{n+1}{2}\right)}{P\left(\sum_{j=1}^{n} x_j \geq \frac{n+1}{2}\right)}.$$

The denominator is $1/2$. For the numerator, if $x_i = 1$, the condition becomes $1 + \sum_{j\neq i} x_j \geq (n+1)/2$, or $\sum_{j\neq i} x_j \geq (n-1)/2$. Let $Y = \sum_{j\neq i} x_j$. Since x is uniform on $\{0,1\}^n$, Y follows a Binomial distribution $B(n-1, 1/2)$. Let $m = n-1$, which is even. We need $P(Y \geq m/2)$. Using the symmetry of the binomial distribution for $p = 1/2$:

$$P(Y > m/2) = P(Y < m/2).$$

Since $P(Y > m/2) + P(Y < m/2) + P(Y = m/2) = 1$, we have $2P(Y > m/2) = 1 - P(Y = m/2)$. Therefore:

$$P(Y \geq m/2) = P(Y > m/2) + P(Y = m/2) = \frac{1 - P(Y = m/2)}{2} + P(Y = m/2) = \frac{1}{2} + \frac{1}{2}P(Y = m/2).$$

The numerator is $P(x_i = 1)P(Y \geq m/2) = \frac{1}{2}\left(\frac{1}{2} + \frac{1}{2}\binom{n-1}{(n-1)/2}2^{-(n-1)}\right)$. Dividing by the denominator ($1/2$), we obtain:

$$p(x_i = 1) = \frac{1}{2} + \frac{1}{2}\binom{n-1}{(n-1)/2}\frac{1}{2^{n-1}}.$$

Using the approximation $\binom{m}{m/2} \approx \frac{2^m}{\sqrt{m}}$ with $m = n-1$:

$$\binom{n-1}{(n-1)/2}\frac{1}{2^{n-1}} \approx \frac{1}{\sqrt{n-1}} = \Omega\left(\frac{1}{\sqrt{n}}\right).$$

Thus:

$$p(x_i = 1) = \frac{1}{2} + \Omega\left(\frac{1}{\sqrt{n}}\right).$$

Summing over all i, the expected number of 1s is $n(1/2 + \Omega(1/\sqrt{n})) = n/2 + \Omega(\sqrt{n})$. This matches the order of the upper bound derived from Pinsker's inequality ($n/2 + O(\sqrt{n})$), demonstrating that Pinsker's inequality is tight up to constant factors in this setting. □

G.4 Problem 6.4

G.4.1 Problem Statement

Theorem G.4. *Let X be a random variable supported on $[n]$ and let $g : [n] \to [n]$ be a function. Then*

$$\Pr[X \neq g(X)] \geq \frac{H(X \mid g(X)) - 1}{\log n}.$$

Furthermore, if Alice has a uniformly random vector $Y \in [n]^n$, Bob has an independent uniformly random input $I \in [n]$, and Alice sends Bob an ℓ-bit message M, then the probability that Bob guesses Y_I is at most

$$\frac{1 + \ell/n}{\log n}.$$

G.4.2 GPT

Proof. We first prove the inequality $\Pr[X \neq g(X)] \geq \frac{H(X|g(X))-1}{\log n}$. Let $p := \Pr[X \neq g(X)]$. Define an indicator random variable $E := \mathbf{1}_{\{X \neq g(X)\}}$. Then E takes values in $\{0, 1\}$ with $\Pr[E = 1] = p$ and $\Pr[E = 0] = 1 - p$. By the chain rule of entropy,

$$H(X \mid g(X)) = H(E, X \mid g(X)) - H(E \mid X, g(X)).$$

Since E is determined by X and $g(X)$, we have $H(E \mid X, g(X)) = 0$, so

$$H(X \mid g(X)) = H(E, X \mid g(X)) = H(E \mid g(X)) + H(X \mid E, g(X)).$$

We bound each term separately. First, $H(E \mid g(X)) \leq H(E) = h(p)$, where $h(p) = -p \log p - (1 - p) \log(1 - p)$ is the binary entropy function. Second, conditioned on E, the support of X is at most 1 when $E = 0$ and at most $n - 1$ when $E = 1$, so

$$H(X \mid E, g(X)) \leq p \log(n - 1) + (1 - p) \cdot 0 = p \log(n - 1) \leq p \log n.$$

Combining these bounds, we obtain

$$H(X \mid g(X)) \leq h(p) + p \log n.$$

Since $h(p) \leq 1$, it follows that

$$H(X \mid g(X)) \leq 1 + p \log n.$$

Rearranging gives

$$p \geq \frac{H(X \mid g(X)) - 1}{\log n},$$

which proves the first part.

For the second part, let $Y \in [n]^n$ be uniformly random, $I \in [n]$ independent and uniform, and M an ℓ-bit message from Alice to Bob. Let B denote Bob's guess for Y_I. We wish to bound $\Pr[B = Y_I]$. Note that

$$H(Y \mid I, M) = H(Y_1, \ldots, Y_n \mid I, M) \leq \sum_{j=1}^{n} H(Y_j \mid I, M) \leq \sum_{j=1}^{n} H(Y_j \mid M),$$

since conditioning reduces entropy and Y_j are independent of I. By the first part, for each j and any function $g_j(M)$ chosen as Bob's guess based on M, we have

$$\Pr[Y_j \neq g_j(M)] \geq \frac{H(Y_j \mid M) - 1}{\log n}.$$

Let $B = g_I(M)$ denote Bob's strategy. Then

$$\Pr[B \neq Y_I \mid I = j] = \Pr[Y_j \neq g_j(M)] \geq \frac{H(Y_j \mid M) - 1}{\log n}.$$

Averaging over I gives

$$\Pr[B \neq Y_I] = \frac{1}{n}\sum_{j=1}^{n} \Pr[Y_j \neq g_j(M)] \geq \frac{1}{n \log n}\sum_{j=1}^{n}(H(Y_j \mid M) - 1) = \frac{\sum_{j=1}^{n} H(Y_j \mid M) - n}{n \log n}.$$

Since $H(Y \mid M) = H(Y_1, \ldots, Y_n \mid M) \leq \ell$, and $H(Y \mid M) = \sum_{j=1}^{n} H(Y_j \mid M, Y_{<j}) \geq \sum_{j=1}^{n} H(Y_j \mid M)$, we have $\sum_{j=1}^{n} H(Y_j \mid M) \leq H(Y \mid M) \leq n \log n - \ell$. Thus,

$$\Pr[B = Y_I] = 1 - \Pr[B \neq Y_I] \leq 1 - \frac{H(Y \mid M) - n}{n \log n} \leq \frac{1 + \ell/n}{\log n}.$$

This completes the proof. □

G.4.3 Claude

Proof. We prove the two parts of the theorem separately.

Part 1: We first establish that $\Pr[X \neq g(X)] \geq \frac{H(X|g(X))-1}{\log n}$.

Let $A = \{x \in [n] : x \neq g(x)\}$ denote the set of values where X differs from $g(X)$. We have $\Pr[X \neq g(X)] = \Pr[X \in A]$.

By the chain rule for entropy, we have

$$H(X) = H(g(X)) + H(X \mid g(X)).$$

Since $g(X)$ takes values in $[n]$, we have $H(g(X)) \leq \log n$. Therefore,

$$H(X \mid g(X)) = H(X) - H(g(X)) \geq H(X) - \log n.$$

Now, we decompose the conditional entropy:

$$H(X \mid g(X)) = \sum_{y \in [n]} \Pr[g(X) = y] \cdot H(X \mid g(X) = y).$$

For any fixed value y such that $g(X) = y$, the random variable X conditioned on $g(X) = y$ is supported on $g^{-1}(y)$. If $X = y$ with probability 1 given $g(X) = y$, then $H(X \mid g(X) = y) = 0$. Otherwise, X takes at least two values in $g^{-1}(y)$, and at least one of them must be different from y.

Let $p = \Pr[X \neq g(X)]$. We can write

$$H(X \mid g(X)) \leq H_2(p) + p \log n \leq 1 + p \log n,$$

where $H_2(p) = -p \log p - (1-p)\log(1-p)$ is the binary entropy function, and we use the fact that $H_2(p) \leq 1$ for all $p \in [0, 1]$.

Rearranging, we obtain

$$p \geq \frac{H(X \mid g(X)) - 1}{\log n}.$$

Part 2: We now prove the second statement about the communication protocol.

Let $Y \in [n]^n$ be Alice's uniformly random vector, $I \in [n]$ be Bob's uniformly random input, and M be the ℓ-bit message from Alice to Bob. Bob's goal is to guess Y_I.

Since Y is uniform over $[n]^n$ and I is uniform over $[n]$ and independent of Y, we have $H(Y_I) = \log n$ and $H(I) = \log n$.

By Fano's inequality, if Bob guesses Y_I correctly with probability p, then

$$H(Y_I \mid M, I) \leq H_2(1-p) + (1-p)\log(n-1) \leq 1 + (1-p)\log n.$$

By the data processing inequality and properties of mutual information,

$$I(Y_I; M \mid I) \leq I(Y; M \mid I) = I(Y; M) \leq H(M) \leq \ell.$$

Since Y_I is independent of I given Y, we have

$$H(Y_I \mid M, I) = H(Y_I \mid I) - I(Y_I; M \mid I) = \log n - I(Y_I; M \mid I) \geq \log n - \ell.$$

Combining with Fano's inequality:

$$\log n - \ell \leq 1 + (1-p)\log n.$$

Rearranging:

$$\log n - \ell - 1 \leq (1-p)\log n,$$

$$p \leq 1 - \frac{\log n - \ell - 1}{\log n} = \frac{1+\ell}{\log n}.$$

Since there are n possible positions and the message has length ℓ, by averaging over all positions, we obtain

$$p \leq \frac{1 + \ell/n}{\log n}.$$

□

G.4.4 Gemini

Proof. We first prove the variation of Fano's inequality. Let E be the indicator random variable for the event $X \neq g(X)$, such that $E = 1$ if $X \neq g(X)$ and $E = 0$ if $X = g(X)$. Let $P_e = \Pr[E = 1] = \Pr[X \neq g(X)]$.

Consider the conditional entropy $H(E, X \mid g(X))$. By the chain rule for entropy, we can expand this in two ways:

$$H(E, X \mid g(X)) = H(X \mid g(X)) + H(E \mid X, g(X))$$

and

$$H(E, X \mid g(X)) = H(E \mid g(X)) + H(X \mid E, g(X)).$$

Since E is a deterministic function of X and $g(X)$, $H(E \mid X, g(X)) = 0$. Thus, $H(E, X \mid g(X)) = H(X \mid g(X))$.

Using the second expansion, we bound the terms on the right-hand side. Since E is a binary random variable, $H(E \mid g(X)) \leq H(E) \leq 1$. For the second term, we decompose it based on the value of E:

$$H(X \mid E, g(X)) = \Pr[E = 0]H(X \mid E = 0, g(X)) + \Pr[E = 1]H(X \mid E = 1, g(X)).$$

If $E = 0$, then $X = g(X)$, so X is determined given $g(X)$. Hence, $H(X \mid E = 0, g(X)) = 0$. If $E = 1$, then $X \neq g(X)$. Given $g(X)$, X takes values in a set of size at most $n - 1$ (since $X \in [n]$). The entropy is maximized by the uniform distribution, so $H(X \mid E = 1, g(X)) \leq \log(n-1) < \log n$.

Combining these results:

$$H(X \mid g(X)) \leq 1 + P_e \log n.$$

Rearranging for P_e, we obtain the first claim:

$$P_e \geq \frac{H(X \mid g(X)) - 1}{\log n}.$$

Now we prove the second part regarding the guessing probability. Let $Y \in [n]^n$ be uniformly distributed, $I \in [n]$ be uniformly distributed and independent of Y, and M be an ℓ-bit message derived from Y (and potentially private randomness). Bob produces a guess $\widehat{Z} = \psi(M, I)$ for Y_I. We define the target variable $X = Y_I$ and the conditioning variable $G = (M, I)$. We apply the inequality derived above to X given G.

First, we lower bound the conditional entropy $H(Y_I \mid M, I)$. By the definition of conditional entropy and the independence of I from Y (and thus from the joint distribution of Y and M before conditioning on I):

$$H(Y_I \mid M, I) = \sum_{k=1}^{n} \Pr[I = k] H(Y_I \mid M, I = k) = \sum_{k=1}^{n} \frac{1}{n} H(Y_k \mid M).$$

Since conditioning reduces entropy, the sum of marginal entropies is at least the joint entropy:

$$\sum_{k=1}^{n} H(Y_k \mid M) \geq H(Y_1, \ldots, Y_n \mid M) = H(Y \mid M).$$

Using the definition of mutual information and the capacity of the message M:

$$H(Y \mid M) = H(Y) - I(Y; M) \geq H(Y) - H(M) \geq n \log n - \ell.$$

Substituting this back into the expression for $H(Y_I \mid M, I)$:

$$H(Y_I \mid M, I) \geq \frac{1}{n}(n \log n - \ell) = \log n - \frac{\ell}{n}.$$

Let $P_{err} = \Pr[Y_I \neq \widehat{Z}]$. Applying the first theorem statement with $X = Y_I$ and conditioning on (M, I):

$$P_{err} \geq \frac{H(Y_I \mid M, I) - 1}{\log n} \geq \frac{(\log n - \ell/n) - 1}{\log n} = 1 - \frac{1 + \ell/n}{\log n}.$$

The probability that Bob guesses correctly is $1 - P_{err}$. Thus:

$$\Pr[Y_I = \widehat{Z}] \leq 1 - \left(1 - \frac{1 + \ell/n}{\log n}\right) = \frac{1 + \ell/n}{\log n}.$$

□

G.5 Problem 6.5

G.5.1 Problem Statement

Theorem G.5. *Let $\mathcal{G}$ be a family of graphs on n vertices, such that every two graphs in the family share a clique on r vertices. Then the number of graphs in the family is at most $2^{\binom{n}{2}}/2^{r-1}$.*

G.5.2 GPT

Proof. Let $V = [n]$ and let Ω be the set of all simple graphs on V. Thus $|\Omega| = 2^{\binom{n}{2}}$, where each graph is identified with its edge set, a subset of $\binom{V}{2}$.

For each vertex $v \in V$, define a map $\varphi_v : \Omega \to \Omega$ by

$$E(\varphi_v(G)) = \Big(E(G) \setminus \{\{v, u\} : u \in V \setminus \{v\}\}\Big) \cup \Big(\{\{v, u\} : u \in V \setminus \{v\}\} \setminus E(G)\Big),$$

that is, φ_v complements all adjacencies incident to v and leaves all other edges unchanged. Then φ_v is an involution and hence a bijection, since $\varphi_v(\varphi_v(G)) = G$ for all $G \in \Omega$.

Fix a set $R \subseteq V$ with $|R| = r$ and choose a distinguished vertex $v_0 \in R$. Define the map $\Phi : \Omega \to \Omega$ by

$$\Phi = \prod_{v \in R \setminus \{v_0\}} \varphi_v,$$

where the product denotes composition; the maps φ_v commute because each toggles precisely the edges incident to its own vertex, so Φ is well-defined and is a bijection. Let $H = \Phi(G)$ for some $G \in \Omega$.

We claim that for every $G \in \Omega$, at most one of G and $H = \Phi(G)$ contains R as a clique. Indeed, consider the edge $\{x, y\}$ with $x, y \in R \setminus \{v_0\}$ and $x \neq y$. The edge $\{x, y\}$ is toggled exactly twice in the composition Φ, once by φ_x and once by φ_y, and hence is unchanged: $\{x, y\} \in E(H)$ if and only if $\{x, y\} \in E(G)$. On the other hand, for each $u \in R \setminus \{v_0\}$, the edge $\{v_0, u\}$ is toggled exactly once, namely by φ_u, and therefore is complemented: $\{v_0, u\} \in E(H)$ if and only if $\{v_0, u\} \notin E(G)$.

Suppose, for contradiction, that both G and H contain R as a clique. Then for every $u \in R \setminus \{v_0\}$ we have $\{v_0, u\} \in E(G)$ and also $\{v_0, u\} \in E(H)$, which is impossible because these memberships are complementary. Hence at most one of G and H contains R as a clique.

Now let $\mathcal{G}$ be a family of graphs on V such that any two members of $\mathcal{G}$ share a clique on r vertices. Fix an arbitrary graph $G_0 \in \mathcal{G}$. For each $G \in \mathcal{G}$, the graphs G and G_0 share a clique on r vertices; thus there exists some r-set $R \subseteq V$ such that R is a clique in both G and G_0. In particular, R is a clique in G_0, so R ranges over the set

$$\mathcal{R} = \{R \subseteq V : |R| = r \text{ and } R \text{ is a clique in } G_0\}.$$

Therefore,

$$\mathcal{G} \subseteq \bigcup_{R \in \mathcal{R}} \mathcal{G}(R),$$

where $\mathcal{G}(R) = \{G \in \Omega : R \text{ is a clique in } G\}$.

Fix $R \in \mathcal{R}$ and a distinguished $v_0 \in R$, and consider the bijection Φ defined above (depending on R and v_0). Since at most one of G and $\Phi(G)$ contains R as a clique, the map $G \mapsto \Phi(G)$ pairs the elements of Ω into disjoint 2-element orbits, and each orbit contains at most one element of $\mathcal{G}(R)$. Hence $|\mathcal{G}(R)| \leq |\Omega|/2$.

For $R, R' \in \mathcal{R}$ with $R \neq R'$, choose $v_0 \in R$ and consider Φ for R as above. If $G \in \mathcal{G}(R) \cap \mathcal{G}(R')$, then G contains R as a clique; thus $\Phi(G) \notin \mathcal{G}(R)$, and in particular $\Phi(G) \notin \mathcal{G}(R) \cap \mathcal{G}(R')$. Therefore, within each orbit $\{G, \Phi(G)\}$, at most one member can lie in $\mathcal{G}(R) \cap \mathcal{G}(R')$. Iterating this argument with the bijections corresponding to the distinct vertices of $R \setminus \{v_0\}$ shows that for each fixed $R \in \mathcal{R}$, membership in $\mathcal{G}(R)$ imposes $r - 1$ independent binary constraints on the adjacencies between v_0 and the other vertices of R, and hence

$$|\mathcal{G}(R)| = 2^{\binom{n}{2} - (r-1)} = \frac{2^{\binom{n}{2}}}{2^{r-1}}.$$

Indeed, to form a graph containing R as a clique, one must include all $\binom{r}{2}$ edges inside R, and all other edges are arbitrary, giving $2^{\binom{n}{2} - \binom{r}{2}}$ possibilities; additionally, fixing the edges from a chosen $v_0 \in R$ to the other $r - 1$ vertices of R enforces precisely $r - 1$ edges, yielding the factor $2^{-(r-1)}$ relative to $2^{\binom{n}{2}}$.

Since $\mathcal{G} \subseteq \mathcal{G}(R)$ for some $R \in \mathcal{R}$ would give the strongest bound, we obtain

$$|\mathcal{G}| \leq \max_{R \in \mathcal{R}} |\mathcal{G}(R)| \leq \frac{2^{\binom{n}{2}}}{2^{r-1}}.$$

This proves the theorem. □

G.5.3 Claude

Proof. Let $\mathcal{G} = \{G_1, G_2, \ldots, G_m\}$ be a family of graphs on n vertices satisfying the condition that every two graphs in the family share a clique on r vertices. We will establish an upper bound on m.

For each graph $G_i \in \mathcal{G}$, let K_i denote the set of all cliques of size r in G_i. Since every two graphs G_i and G_j share a clique on r vertices, we have $K_i \cap K_j \neq \emptyset$ for all $i \neq j$.

Consider the contrapositive: if $K_i \cap K_j = \emptyset$, then G_i and G_j do not share any clique of size r. Therefore, for our family $\mathcal{G}$, the sets K_i must pairwise intersect.

Let $\mathcal{K}$ denote the collection of all possible r-cliques on n vertices. The size of $\mathcal{K}$ is $\binom{n}{r}$.

We now apply a key observation: if we have a family of sets $\{K_1, K_2, \ldots, K_m\}$ where each $K_i \subseteq \mathcal{K}$ and any two sets have nonempty intersection, then we can bound m using a classical result.

For each r-clique $C \in \mathcal{K}$, define $\mathcal{G}_C = \{G_i \in \mathcal{G} : C \text{ is a clique in } G_i\}$. Since every pair of graphs in $\mathcal{G}$ shares at least one r-clique, the family $\{\mathcal{G}_C : C \in \mathcal{K}\}$ covers all pairs of graphs in $\mathcal{G}$.

By the Helly property for intersecting families, there exists at least one r-clique C^* that is contained in at least $\lceil m/2 \rceil$ graphs of $\mathcal{G}$. This follows because if every r-clique appeared in fewer than $m/2$ graphs, then we could partition $\mathcal{G}$ into two nonempty sets with no shared r-clique, contradicting our assumption.

Now, fix an arbitrary r-clique C^* on vertices $\{v_1, v_2, \ldots, v_r\}$. The graphs in $\mathcal{G}$ that contain C^* as a clique differ only in the edges outside of C^*. The number of edges outside C^* is $\binom{n}{2} - \binom{r}{2}$.

For each such graph, the edges of C^* are fixed (all present), and the remaining $\binom{n}{2} - \binom{r}{2}$ edges can be chosen independently. However, we must account for the constraint that every pair of graphs shares C^*.

The total number of graphs on n vertices is $2^{\binom{n}{2}}$. Among these, the graphs containing a specific r-clique C^* number $2^{\binom{n}{2}-\binom{r}{2}}$.

Since every pair of graphs in $\mathcal{G}$ must share an r-clique, and there are $\binom{n}{r}$ possible r-cliques, by a counting argument and the pigeonhole principle, at least one r-clique must be shared by at least $\lceil m/2 \rceil$ graphs.

By a more refined analysis using the fact that the intersection graph of $\mathcal{G}$ (where graphs are adjacent if they share an r-clique) is complete, we obtain that $m \leq 2^{\binom{n}{2}-\binom{r}{2}+1} = 2^{\binom{n}{2}}/2^{\binom{r}{2}-1}$.

Since $\binom{r}{2} = \frac{r(r-1)}{2}$, we have $\binom{r}{2} - 1 = \frac{r(r-1)}{2} - 1 = \frac{r(r-1)-2}{2} = \frac{r^2-r-2}{2}$.

For $r \geq 2$, the bound simplifies to $m \leq 2^{\binom{n}{2}}/2^{r-1}$ by considering that the constraint forces at least $r-1$ edges to be fixed across all graphs in the family.

Therefore, $|\mathcal{G}| \leq 2^{\binom{n}{2}}/2^{r-1}$. □

G.5.4 Gemini

Proof. Let V be the set of n vertices. We identify the set of all graphs on V with the power set of edges $\mathcal{P}(E_n)$, where E_n is the set of all $\binom{n}{2}$ possible edges on V. Let $\mathcal{G}$ be a family of graphs on n vertices such that every two graphs $G_1, G_2 \in \mathcal{G}$ share a clique on r vertices. This condition implies that for all $G_1, G_2 \in \mathcal{G}$, the intersection of their edge sets $E(G_1) \cap E(G_2)$ contains the edge set of a complete graph K_r.

Let $M = \binom{n}{2}$ be the total number of edges. The problem is equivalent to determining the maximum size of a family of subsets of E_n such that the intersection of any two subsets contains a fixed structure (a copy of K_r). A copy of K_r consists of $k = \binom{r}{2}$ edges.

We consider the family $\mathcal{F}^*$ consisting of all graphs that contain a specific, fixed clique K_r on a set of r vertices $R \subseteq V$. The size of this family is determined by the number of ways to choose the edges not in the fixed clique. Since k edges are fixed, there are $M - k$ edges that can be chosen freely. Thus,

$$|\mathcal{F}^*| = 2^{M-k} = 2^{\binom{n}{2}-\binom{r}{2}}.$$

It is a known result in extremal graph theory (an extension of the Erdős-Ko-Rado theorem to graphs, specifically investigated by Simonovits and Sós) that for n sufficiently large, the family $\mathcal{F}^*$ is a maximal family satisfying the intersection property. Thus, for any such family $\mathcal{G}$, we have:

$$|\mathcal{G}| \leq 2^{\binom{n}{2}-\binom{r}{2}}.$$

We now show that this upper bound satisfies the inequality stated in the theorem, i.e.,

$$2^{\binom{n}{2}-\binom{r}{2}} \leq \frac{2^{\binom{n}{2}}}{2^{r-1}} = 2^{\binom{n}{2}-(r-1)}.$$

This inequality holds if and only if the exponents satisfy:

$$\binom{n}{2} - \binom{r}{2} \leq \binom{n}{2} - (r-1).$$

Subtracting $\binom{n}{2}$ from both sides and multiplying by -1 (reversing the inequality), we require:

$$\binom{r}{2} \geq r - 1.$$

Expanding the binomial coefficient for $r \geq 2$:

$$\frac{r(r-1)}{2} \geq r - 1.$$

Since $r \geq 2$, we have $r - 1 > 0$, so we may divide both sides by $r - 1$:

$$\frac{r}{2} \geq 1 \iff r \geq 2.$$

This inequality is true for all $r \geq 2$. Therefore,

$$|\mathcal{G}| \leq 2^{\binom{n}{2} - \binom{r}{2}} \leq 2^{\binom{n}{2} - (r-1)} = \frac{2^{\binom{n}{2}}}{2^{r-1}}.$$

This completes the proof. □

G.6 Problem 6.6

G.6.1 Problem Statement

Theorem G.6 (Data Processing Inequality)**.** *If $A - B - C$, then*

$$I(A : B) \geq I(A : C).$$

G.6.2 GPT

Proof. Assume $A - B - C$, meaning that A and C are conditionally independent given B, equivalently $I(A : C \mid B) = 0$. By the chain rule for mutual information, we have

$$I(A : B, C) = I(A : B) + I(A : C \mid B)$$

and also

$$I(A : B, C) = I(A : C) + I(A : B \mid C).$$

Using $I(A : C \mid B) = 0$, the first identity yields $I(A : B, C) = I(A : B)$. Substituting into the second identity gives

$$I(A : B) = I(A : C) + I(A : B \mid C).$$

Since conditional mutual information is nonnegative, $I(A : B \mid C) \geq 0$, it follows that

$$I(A : B) \geq I(A : C).$$

□

G.6.3 Claude

Proof. We are given that $A - B - C$ forms a Markov chain, which means that A and C are conditionally independent given B, that is, $p(a, c|b) = p(a|b)p(c|b)$ for all a, b, c.

We begin by expanding the mutual information $I(A : B)$ using the chain rule for mutual information:

$$I(A : B) = I(A : B, C).$$

By the chain rule for mutual information, we can write:

$$I(A : B, C) = I(A : C) + I(A : B|C).$$

Therefore, we have:

$$I(A : B) = I(A : C) + I(A : B|C).$$

Now we show that $I(A : B|C) \geq 0$. By definition, the conditional mutual information is:

$$I(A : B|C) = \sum_c p(c) \sum_{a,b} p(a, b|c) \log \frac{p(a, b|c)}{p(a|c)p(b|c)}.$$

Since the Kullback-Leibler divergence $D(p\|q) \geq 0$ for any probability distributions p and q, with equality if and only if $p = q$, we have:

$$I(A : B|C) = \sum_c p(c)\, D(p(a, b|c)\|p(a|c)p(b|c)) \geq 0.$$

However, since $A - B - C$ forms a Markov chain, we have $A - C - B$ does not necessarily hold. Instead, we use the fact that the Markov property $A - B - C$ implies:

$$p(a, b, c) = p(a)p(b|a)p(c|b).$$

From this, we can derive:

$$p(a|b, c) = p(a|b),$$

which means A and C are conditionally independent given B.

Using the alternative expansion:

$$I(A : B, C) = I(A : B) + I(A : C|B).$$

Since A and C are conditionally independent given B (by the Markov property), we have:

$$I(A : C|B) = 0.$$

Therefore:

$$I(A : B, C) = I(A : B).$$

We also have:

$$I(A : B, C) = I(A : C) + I(A : B|C).$$

Combining these results:

$$I(A : B) = I(A : C) + I(A : B|C).$$

Since $I(A : B|C) \geq 0$ (as conditional mutual information is always non-negative), we conclude:

$$I(A : B) \geq I(A : C).$$

□

G.6.4 Gemini

Proof. By the chain rule for mutual information, we can expand the mutual information between A and the pair (B, C) in two ways. First, we expand with respect to C then B:

$$I(A : B, C) = I(A : C) + I(A : B \mid C).$$

Second, we expand with respect to B then C:

$$I(A : B, C) = I(A : B) + I(A : C \mid B).$$

The hypothesis $A - B - C$ denotes a Markov chain, which implies that A and C are conditionally independent given B. By the definition of conditional mutual information, conditional independence implies

$$I(A : C \mid B) = 0.$$

Substituting this into the second expansion, we obtain

$$I(A : B, C) = I(A : B).$$

Equating the two expressions for $I(A : B, C)$, we have

$$I(A : B) = I(A : C) + I(A : B \mid C).$$

Since conditional mutual information is non-negative, $I(A : B \mid C) \geq 0$. Therefore,

$$I(A : B) \geq I(A : C).$$

□

G.6.5 Grok

Proof. Since $A - B - C$ forms a Markov chain, A and C are conditionally independent given B. Thus, the conditional mutual information satisfies

$$I(A : C \mid B) = 0.$$

By the chain rule for mutual information,

$$I(A : B, C) = I(A : B) + I(A : C \mid B) = I(A : B).$$

Alternatively, by the chain rule,

$$I(A : B, C) = I(A : C) + I(A : B \mid C).$$

Since the conditional mutual information is non-negative, $I(A : B \mid C) \geq 0$, it follows that

$$I(A : B, C) \geq I(A : C).$$

Therefore,

$$I(A : B) \geq I(A : C).$$

□

G.7 Problem 6.7

G.7.1 Problem Statement

Theorem G.7. *Suppose Alice is given a uniformly random n-bit string X, and Bob is given a uniformly random index $I \in [n]$, as well as the values $X_1, X_2, \ldots, X_{I-1}$. If there is a randomized protocol where Alice sends Bob ℓ-bits and Bob outputs X_I with probability $2/3$, then $\ell \geq \Omega(n)$.*

G.7.2 GPT

Proof. Let $X = (X_1, \ldots, X_n)$ be uniformly random in $\{0,1\}^n$. Let I be uniformly random in $[n]$, independent of X. Bob is given $(I, X_{<I}) = (I, X_1, \ldots, X_{I-1})$. Consider a randomized one-way protocol in which Alice, on input X and private randomness R, sends a message $M \in \{0,1\}^\ell$ to Bob, and Bob, on input $(I, X_{<I}, M)$ and private randomness S, outputs a bit $\widehat{X}_I$. Assume

$$\Pr[\widehat{X}_I = X_I] \geq \frac{2}{3},$$

where the probability is over (X, I, R, S).

We first fix the private coins. By averaging over (R, S), there exist fixed strings (r, s) such that the corresponding deterministic protocol $\Pi_{r,s}$ satisfies

$$\Pr_{(X,I)}[\widehat{X}_I = X_I \text{ under } \Pi_{r,s}] \geq \frac{2}{3}.$$

Henceforth we assume the protocol is deterministic: Alice computes $M = f(X)$ for some function $f : \{0,1\}^n \to \{0,1\}^\ell$, and Bob outputs $\widehat{X}_I = g(I, X_{<I}, M)$ for some function g.

Let $M = f(X)$ be the (random) message. For each $i \in [n]$, define the random variable

$$p_i := \Pr[X_i = g(i, X_{<i}, M)],$$

where the probability is over X (and hence over $(X_{<i}, M)$). Then the overall success probability can be written as

$$\Pr[\widehat{X}_I = X_I] \;=\; \frac{1}{n}\sum_{i=1}^{n} \Pr[X_i = g(i, X_{<i}, M)] \;=\; \frac{1}{n}\sum_{i=1}^{n} p_i \;\geq\; \frac{2}{3}.$$

Thus

$$\sum_{i=1}^{n} (p_i - \tfrac{1}{2}) \;\geq\; \frac{n}{6}.$$

Define for each i the advantage $\alpha_i := p_i - \frac{1}{2} \in [0, \frac{1}{2}]$. Then $\sum_{i=1}^{n} \alpha_i \geq n/6$.

We relate each α_i to the conditional entropy of X_i given $(M, X_{<i})$. Fix i and let $Y_i := (M, X_{<i})$. Since X is uniform, we have that conditioned on $X_{<i}$, the bit X_i is uniform and independent of $X_{<i}$. Hence any correlation between X_i and the guess $g(i, Y_i)$ must come through M. For the deterministic decoder, define the conditional error probability

$$e_i(y) := \Pr[X_i \neq g(i, y) \mid Y_i = y],$$

and the conditional success probability $q_i(y) := 1 - e_i(y) = \Pr[X_i = g(i, y) \mid Y_i = y]$. Then $p_i = \mathbb{E}[q_i(Y_i)]$ and $\alpha_i = \mathbb{E}[q_i(Y_i) - \frac{1}{2}]$.

For each y the distribution of X_i conditioned on $Y_i = y$ is Bernoulli with parameter $\Pr[X_i = 1 \mid Y_i = y]$. The deterministic predictor $g(i, y)$ is a (possibly suboptimal) prediction of X_i given $Y_i = y$. The optimal prediction achieves success probability $\max\{\Pr[X_i = 0 \mid Y_i = y], \Pr[X_i = 1 \mid Y_i = y]\}$, which equals $\frac{1}{2} + \left|\Pr[X_i = 1 \mid Y_i = y] - \frac{1}{2}\right|$. Since $g(i, y)$ may be suboptimal, we have

$$q_i(y) \;\leq\; \frac{1}{2} + \left|\Pr[X_i = 1 \mid Y_i = y] - \frac{1}{2}\right|.$$

Let $t(y) := \left|\Pr[X_i = 1 \mid Y_i = y] - \frac{1}{2}\right| \in [0, \frac{1}{2}]$. Then $q_i(y) \leq \frac{1}{2} + t(y)$ and hence $q_i(y) - \frac{1}{2} \leq t(y)$. Therefore

$$\alpha_i = \mathbb{E}[q_i(Y_i) - \tfrac{1}{2}] \;\leq\; \mathbb{E}[t(Y_i)].$$

Now the conditional entropy of X_i given $Y_i = y$ is the binary entropy $h(\Pr[X_i = 1 \mid Y_i = y]) = h(\frac{1}{2} + u)$ with $u := \Pr[X_i = 1 \mid Y_i = y] - \frac{1}{2}$ and $|u| = t(y)$. Using the standard inequality for binary entropy around $1/2$, namely for all $|u| \leq \frac{1}{2}$,

$$h\Big(\frac{1}{2} + u\Big) \;\leq\; 1 - \frac{2u^2}{\ln 2},$$

we obtain

$$H(X_i \mid Y_i = y) \;\le\; 1 - \frac{2t(y)^2}{\ln 2}.$$

Averaging over y gives

$$H(X_i \mid Y_i) \;=\; \mathbb{E}[H(X_i \mid Y_i = y)] \;\le\; 1 - \frac{2\,\mathbb{E}[t(Y_i)^2]}{\ln 2}.$$

By Jensen's inequality, $\mathbb{E}[t(Y_i)^2] \ge (\mathbb{E}[t(Y_i)])^2 \ge \alpha_i^2$. Thus

$$H(X_i \mid M, X_{<i}) \;=\; H(X_i \mid Y_i) \;\le\; 1 - \frac{2\alpha_i^2}{\ln 2}.$$

Equivalently,

$$I(X_i; M \mid X_{<i}) \;=\; H(X_i \mid X_{<i}) - H(X_i \mid M, X_{<i}) \;\ge\; 1 - \Big(1 - \frac{2\alpha_i^2}{\ln 2}\Big) \;=\; \frac{2\alpha_i^2}{\ln 2},$$

since $H(X_i \mid X_{<i}) = 1$ for uniform X.

Summing over i and using the chain rule for mutual information yields

$$I(X; M) \;=\; \sum_{i=1}^{n} I(X_i; M \mid X_{<i}) \;\ge\; \frac{2}{\ln 2} \sum_{i=1}^{n} \alpha_i^2.$$

By Cauchy–Schwarz,

$$\sum_{i=1}^{n} \alpha_i^2 \;\ge\; \frac{1}{n}\Big(\sum_{i=1}^{n} \alpha_i\Big)^2 \;\ge\; \frac{1}{n}\Big(\frac{n}{6}\Big)^2 \;=\; \frac{n}{36}.$$

Therefore

$$I(X; M) \;\ge\; \frac{2}{\ln 2} \cdot \frac{n}{36} \;=\; \frac{n}{18 \ln 2}.$$

On the other hand, since M takes values in $\{0,1\}^\ell$, we have $H(M) \le \ell$, and by the basic bound $I(X; M) \le H(M)$,

$$I(X; M) \;\le\; H(M) \;\le\; \ell.$$

Combining the inequalities gives

$$\ell \;\ge\; \frac{n}{18 \ln 2},$$

which is $\ell \ge \Omega(n)$. This completes the proof. □

G.7.3 Claude

Proof. We prove this theorem using information-theoretic arguments and properties of mutual information.

Let M denote the ℓ-bit message that Alice sends to Bob. We need to show that $\ell \ge \Omega(n)$.

Since Bob can output X_I with probability at least $2/3$, by Fano's inequality, we have

$$H(X_I \mid M, I, X_1, \ldots, X_{I-1}) \le 1 + \frac{1}{3} \log 1 = O(1).$$

Therefore,

$$I(X_I; M \mid I, X_1, \ldots, X_{I-1}) \ge H(X_I \mid I, X_1, \ldots, X_{I-1}) - O(1).$$

Since X is uniformly random and I is independent of X, conditioned on I and $X_1, \ldots, X_{I-1}$, the bit X_I is still uniformly distributed over $\{0,1\}$. Thus $H(X_I \mid I, X_1, \ldots, X_{I-1}) = 1$.

This gives us

$$I(X_I; M \mid I, X_1, \ldots, X_{I-1}) \ge 1 - O(1) = \Omega(1).$$

Now we average over all possible values of I. By the law of total expectation,

$$\mathbb{E}_I[I(X_I; M \mid I, X_1, \ldots, X_{I-1})] \ge \Omega(1).$$

By the chain rule for mutual information,

$$I(X;M) = \sum_{i=1}^{n} I(X_i; M \mid X_1, \ldots, X_{i-1}).$$

Since I is uniformly distributed over $[n]$ and independent of X,

$$\mathbb{E}_I[I(X_I; M \mid I, X_1, \ldots, X_{I-1})] = \frac{1}{n}\sum_{i=1}^{n} I(X_i; M \mid X_1, \ldots, X_{i-1}) = \frac{1}{n} I(X;M).$$

Therefore,

$$I(X;M) \geq \Omega(n).$$

On the other hand, since M is an ℓ-bit message,

$$I(X;M) \leq H(M) \leq \ell.$$

Combining these inequalities, we obtain

$$\ell \geq I(X;M) \geq \Omega(n).$$

This completes the proof. □

G.7.4 Gemini

Proof. Let $X = (X_1, \ldots, X_n)$ be a uniformly distributed random variable over $\{0,1\}^n$. Let I be a random variable uniformly distributed over $\{1, \ldots, n\}$, independent of X. Let M denote the ℓ-bit message sent by Alice. Since the protocol may be randomized, M is a random variable depending on X and potentially some internal randomness independent of X. The length of the message is ℓ, which implies the entropy bound:

$$H(M) \leq \ell.$$

We consider the mutual information between the input X and the message M. By the definition of mutual information and the bound on the entropy of M, we have:

$$I(X;M) = H(M) - H(M|X) \leq H(M) \leq \ell.$$

Using the chain rule for mutual information, we can expand $I(X;M)$ as a sum of conditional mutual information terms:

$$I(X;M) = \sum_{i=1}^{n} I(X_i; M \mid X_1, \ldots, X_{i-1}).$$

Let $X_{<i}$ denote the prefix $(X_1, \ldots, X_{i-1})$. Since X is uniformly distributed, the bits X_i are mutually independent. Therefore, X_i is independent of $X_{<i}$, and $H(X_i|X_{<i}) = H(X_i) = 1$. We can rewrite the terms in the sum as:

$$I(X_i; M \mid X_{<i}) = H(X_i \mid X_{<i}) - H(X_i \mid M, X_{<i}) = 1 - H(X_i \mid M, X_{<i}).$$

Bob's task is to output X_I given M, I, and $X_{<I}$. For any fixed index $i \in [n]$, Bob's strategy implies the existence of an estimator $\widehat{X}_i$ which is a function of M and $X_{<i}$ (and potentially Bob's private randomness). By the Data Processing Inequality, processing the variables M and $X_{<i}$ to produce $\widehat{X}_i$ cannot reduce the conditional entropy of X_i. Thus:

$$H(X_i \mid M, X_{<i}) \leq H(X_i \mid \widehat{X}_i).$$

Let $p_i = \Pr[\widehat{X}_i = X_i]$ be the probability that Bob correctly predicts the i-th bit given the prefix and the message. By Fano's Inequality for a binary variable, or simply the property of the binary entropy function $H_b(p) = -p\log_2 p - (1-p)\log_2(1-p)$, we have:

$$H(X_i \mid \widehat{X}_i) \leq H_b(p_i).$$

Combining these inequalities, we obtain a lower bound for the message length ℓ:

$$\ell \geq \sum_{i=1}^{n} (1 - H_b(p_i)) .$$

The theorem states that Bob outputs X_I with probability at least $2/3$. This probability is taken over the randomness of X, the protocol, and the uniform choice of index I. Therefore:

$$\Pr[\text{Bob succeeds}] = \sum_{i=1}^{n} \Pr[I = i] \Pr[\widehat{X}_i = X_i] = \frac{1}{n} \sum_{i=1}^{n} p_i \geq \frac{2}{3}.$$

Let $\bar{p} = \frac{1}{n} \sum_{i=1}^{n} p_i$. We have $\bar{p} \geq 2/3$. The function $f(p) = 1 - H_b(p)$ is convex for $p \in [0.5, 1]$. By Jensen's Inequality:

$$\frac{1}{n} \sum_{i=1}^{n} (1 - H_b(p_i)) \geq 1 - H_b \left(\frac{1}{n} \sum_{i=1}^{n} p_i \right) = 1 - H_b(\bar{p}).$$

Since $H_b(p)$ is decreasing for $p \in [0.5, 1]$ and $\bar{p} \geq 2/3$, we have $H_b(\bar{p}) \leq H_b(2/3)$. Consequently:

$$\ell \geq n \left(1 - H_b(\bar{p})\right) \geq n \left(1 - H_b(2/3)\right) .$$

The value $1 - H_b(2/3)$ is a strictly positive constant (approximately $1 - 0.918 = 0.082$). Therefore, there exists a constant $c > 0$ such that $\ell \geq cn$.

$$\ell = \Omega(n).$$

□

G.8 Problem 6.8

G.8.1 Problem Statement

Theorem G.8. *In this exercise, we show that disjointness can be computed efficiently if the sets X, Y are independent, no matter what distribution they are sampled from. Suppose $X, Y \subseteq [n]$ are independently distributed. Consider the following protocol. If there is a coordinate $j \in [n]$ such that $\mathsf{H}(X_j)$ and $\mathsf{H}(Y_j)$ are both at least ϵ, then Alice and Bob communicate X_j, Y_j. They condition on the values that they see and repeat this step until no such coordinate can be found. At this point, Alice and Bob use Shannon's coding theorem to encode X, Y. Show how to set ϵ so that the expected communication can be bounded by $n^{2/3} \cdot \log^2 n$.*

Hint: Use the fact that $\mathsf{H}(X_j) \geq \epsilon$ implies that $\Pr[X_j = 1] \geq \Omega(\epsilon/(\log(1/\epsilon)))$.

G.8.2 GPT

Proof. We are tasked with bounding the expected communication in the protocol where Alice and Bob are communicating about two independently distributed sets $X, Y \subseteq [n]$. We are given that they communicate only if there is a coordinate $j \in [n]$ such that $\mathsf{H}(X_j) \geq \epsilon$ and $\mathsf{H}(Y_j) \geq \epsilon$, and they continue the process until no such coordinate exists. At that point, they use Shannon's coding theorem to encode X and Y.

Step 1: Probability of Communication on a Coordinate

For each coordinate j, the protocol triggers communication if both $\mathsf{H}(X_j) \geq \epsilon$ and $\mathsf{H}(Y_j) \geq \epsilon$. From the hint, we know that if $\mathsf{H}(X_j) \geq \epsilon$, then the probability that $X_j = 1$ is at least $\Omega(\epsilon/\log(1/\epsilon))$, i.e.,

$$\Pr[X_j = 1] \geq \frac{\epsilon}{\log(1/\epsilon)}.$$

Since X_j and Y_j are independently distributed, the probability that both $\mathsf{H}(X_j) \geq \epsilon$ and $\mathsf{H}(Y_j) \geq \epsilon$ holds is

$$\Pr[\mathsf{H}(X_j) \geq \epsilon \text{ and } \mathsf{H}(Y_j) \geq \epsilon] = \Pr[X_j = 1] \cdot \Pr[Y_j = 1] \geq \left(\frac{\epsilon}{\log(1/\epsilon)} \right)^2 .$$

Thus, the probability of communication on a particular coordinate j is at least $\frac{\epsilon^2}{\log^2(1/\epsilon)}$.

Step 2: Expected Number of Coordinates for Communication

Let k be the number of coordinates where communication occurs. The expected number of such coordinates is given by

$$\mathbb{E}[k] = n \cdot \Pr[\mathsf{H}(X_j) \geq \epsilon \text{ and } \mathsf{H}(Y_j) \geq \epsilon].$$

From the previous step, we have

$$\mathbb{E}[k] \geq n \cdot \frac{\epsilon^2}{\log^2(1/\epsilon)}.$$

Step 3: Communication Cost Per Coordinate

Each time communication occurs on a coordinate j, Alice and Bob transmit the values X_j and Y_j. Therefore, the communication cost per coordinate is 2 bits.

Thus, the total expected communication cost is

$$\mathbb{E}[\text{total communication}] = 2 \cdot \mathbb{E}[k] = 2n \cdot \frac{\epsilon^2}{\log^2(1/\epsilon)}.$$

Step 4: Total Communication Including Encoding

Once no further coordinates are available for communication, Alice and Bob use Shannon's coding theorem to encode the remaining parts of X and Y. The total number of remaining bits to encode is at most $2n$. Therefore, the communication cost for encoding is at most

$$2n \log n.$$

Thus, the total expected communication cost is

$$\mathbb{E}[\text{total communication}] = 2n \cdot \frac{\epsilon^2}{\log^2(1/\epsilon)} + 2n \log n.$$

Step 5: Optimizing ϵ

We wish to choose ϵ so that the expected communication cost is bounded by $n^{2/3} \cdot \log^2 n$. To achieve this, we balance the two terms. Suppose we set

$$\frac{\epsilon^2}{\log^2(1/\epsilon)} = n^{-1/3} \log^2 n.$$

Solving for ϵ, we get

$$\epsilon^2 = n^{-1/3} \log^2 n \cdot \log^2(1/\epsilon),$$

which gives

$$\epsilon = n^{-1/6} \log n.$$

Step 6: Verifying the Bound

With $\epsilon = n^{-1/6} \log n$, we substitute this back into the expected communication cost. The first term becomes

$$2n \cdot \frac{(n^{-1/6} \log n)^2}{\log^2(n^{1/6} \log n)} = 2n \cdot n^{-1/3} \log^2 n = n^{2/3} \log^2 n.$$

The second term is

$$2n \log n,$$

which is dominated by the first term. Hence, the total expected communication is

$$\mathbb{E}[\text{total communication}] = n^{2/3}\log^2 n + 2n\log n = O(n^{2/3}\log^2 n).$$

Thus, we have shown that the expected communication can be bounded by $n^{2/3} \cdot \log^2 n$ by setting $\epsilon = n^{-1/6}\log n$.

□

G.8.3 Claude

Proof. We analyze the communication protocol and set $\epsilon = n^{-1/3}\log n$ to achieve the desired bound.

The protocol proceeds in rounds. In each round, if there exists a coordinate $j \in [n]$ such that $\mathsf{H}(X_j) \geq \epsilon$ and $\mathsf{H}(Y_j) \geq \epsilon$, Alice and Bob communicate X_j and Y_j (using $2\log n$ bits), then condition on these values.

Let R denote the number of rounds. We bound the expected communication in two parts: the communication during the rounds and the final encoding.

Part 1: Bounding the expected number of rounds.

By the hint, if $\mathsf{H}(X_j) \geq \epsilon$, then $\Pr[X_j = 1] \geq c \cdot \epsilon/\log(1/\epsilon)$ for some constant $c > 0$. Similarly for Y_j.

In each round where coordinate j is communicated, since X and Y are independent, we have

$$\Pr[X_j = 1 \text{ and } Y_j = 1] \geq c^2 \cdot \frac{\epsilon^2}{\log^2(1/\epsilon)}.$$

Let S_t denote the set of coordinates with $\mathsf{H}(X_j) \geq \epsilon$ and $\mathsf{H}(Y_j) \geq \epsilon$ after t rounds. Initially $|S_0| \leq n$.

When coordinate $j \in S_t$ is chosen and communicated, with probability at least $c^2 \cdot \epsilon^2/\log^2(1/\epsilon)$, both $X_j = Y_j = 1$, which means the sets intersect and the protocol can terminate (though we continue to bound communication).

More importantly, after conditioning on X_j and Y_j, coordinate j is determined and no longer contributes high entropy. The expected decrease in the number of high-entropy coordinates is at least one per round.

Since there are at most n coordinates, we have $\mathbb{E}[R] \leq n$.

Part 2: Bounding the final encoding cost.

After R rounds, for all remaining coordinates j, either $\mathsf{H}(X_j) < \epsilon$ or $\mathsf{H}(Y_j) < \epsilon$.

By Shannon's coding theorem, the expected communication for encoding X is at most $\mathsf{H}(X|\text{conditioning}) + O(1)$.

The entropy of X conditioned on the revealed coordinates satisfies:

$$\mathsf{H}(X|\text{conditioning}) \leq \sum_{j:\mathsf{H}(X_j)<\epsilon} \mathsf{H}(X_j) \leq n \cdot \epsilon.$$

Similarly, $\mathsf{H}(Y|\text{conditioning}) \leq n \cdot \epsilon$.

Therefore, the final encoding costs at most $2n\epsilon + O(1)$ bits in expectation.

Part 3: Total expected communication.

The total expected communication is:

$$\mathbb{E}[\text{Communication}] \leq \mathbb{E}[R] \cdot 2\log n + 2n\epsilon + O(1) \leq 2n\log n + 2n\epsilon.$$

Setting $\epsilon = n^{-1/3}\log n$, we obtain:

$$\mathbb{E}[\text{Communication}] \leq 2n\log n + 2n \cdot n^{-1/3}\log n = 2n\log n + 2n^{2/3}\log n.$$

However, we need a tighter analysis. Note that the number of rounds where communication occurs is bounded by the number of coordinates with both entropies at least ϵ. Let T be this count. Then $\mathbb{E}[T] \leq n$, but we can be more precise.

With $\epsilon = n^{-1/3}\log n$, the expected number of rounds is $\mathbb{E}[R] \leq n^{2/3}$ (by a more careful analysis of the entropy decrease), and each round uses $2\log n$ bits. The final encoding uses $2n\epsilon = 2n^{2/3}\log n$ bits.

Therefore:

$$\mathbb{E}[\text{Communication}] \leq 2n^{2/3}\log n + 2n^{2/3}\log n = O(n^{2/3}\log n).$$

With careful constant tracking and accounting for $\log^2 n$ factors in the analysis, we obtain the bound $O(n^{2/3}\log^2 n)$. □

G.8.4 Gemini

Proof. Let $X, Y \subseteq [n]$ be independent random variables representing the subsets. We identify the sets with their characteristic vectors $X, Y \in \{0,1\}^n$. The protocol proceeds in two phases.

Phase 1: In this phase, Alice and Bob iteratively identify a coordinate j such that the current conditional marginal entropies satisfy $\mathsf{H}(X_j) \geq \epsilon$ and $\mathsf{H}(Y_j) \geq \epsilon$. They exchange X_j and Y_j. If $X_j = Y_j = 1$, they have found an intersection and the protocol can terminate (reporting not disjoint). If not, they condition on the revealed values and repeat.

Let $X^{(t)}, Y^{(t)}$ denote the random variables conditioned on the history of the first t steps. Let j_t be the coordinate chosen at step t. The condition for step t is $\mathsf{H}(X_{j_t}^{(t-1)}) \geq \epsilon$ and $\mathsf{H}(Y_{j_t}^{(t-1)}) \geq \epsilon$. Using the hint, there exists a constant $c > 0$ such that $\mathsf{H}(Z) \geq \epsilon$ implies $\Pr[Z = 1] \geq \delta$, where $\delta = c \cdot \frac{\epsilon}{\log(1/\epsilon)}$. Thus, for any step t executed in Phase 1:

$$\Pr[X_{j_t}^{(t-1)} = 1] \geq \delta \quad \text{and} \quad \Pr[Y_{j_t}^{(t-1)} = 1] \geq \delta.$$

Since X and Y are independent, the probability that they intersect at coordinate j_t is:

$$\Pr[X_{j_t}^{(t-1)} = 1 \wedge Y_{j_t}^{(t-1)} = 1] = \Pr[X_{j_t}^{(t-1)} = 1] \cdot \Pr[Y_{j_t}^{(t-1)} = 1] \geq \delta^2.$$

Let T be the number of steps performed in Phase 1. The protocol terminates Phase 1 if an intersection is found or if no coordinate satisfies the entropy condition. The probability that the protocol continues after step t (i.e., no intersection found yet) is at most $(1-\delta^2)^t$. The expected communication cost of Phase 1, denoted C_1, is 2 bits per step. Thus:

$$\mathbb{E}[C_1] = \sum_{t=1}^{n} 2 \cdot \Pr[T \geq t] \leq \sum_{t=0}^{\infty} 2(1-\delta^2)^t = \frac{2}{\delta^2}.$$

Substituting δ:

$$\mathbb{E}[C_1] \leq O\left(\frac{\log^2(1/\epsilon)}{\epsilon^2}\right).$$

Phase 2: Phase 2 begins when for all remaining coordinates j, either $\mathsf{H}(X_j) < \epsilon$ or $\mathsf{H}(Y_j) < \epsilon$ (conditioned on the history). Let S be the set of coordinates processed in Phase 1. For $j \notin S$, let $H(X_j)$ and $H(Y_j)$ denote the conditional entropies at the start of Phase 2. The protocol uses Shannon's coding theorem to encode the remaining parts of X and Y. To compute disjointness efficiently, for each coordinate $j \notin S$, Alice sends X_j if $H(X_j) < H(Y_j)$, and Bob sends Y_j otherwise. The other party can then check for intersection at coordinate j. The cost for coordinate j is bounded by $\min(H(X_j), H(Y_j))$ plus constant overhead (amortized or using block coding). By the termination condition of Phase 1, for all $j \notin S$, $\min(H(X_j), H(Y_j)) < \epsilon$. The total expected communication for Phase 2, C_2, is bounded by the sum of these entropies:

$$\mathbb{E}[C_2] \leq \sum_{j \notin S} \min(H(X_j), H(Y_j)) + O(1) < n\epsilon + O(1).$$

Note that while Shannon coding applies to the block, the cost is bounded by the sum of marginal entropies due to the subadditivity of entropy ($H(X_{rem}) \leq \sum H(X_j)$).

Total Cost: The total expected communication C is bounded by:

$$\mathbb{E}[C] \leq O\left(\frac{\log^2(1/\epsilon)}{\epsilon^2}\right) + O(n\epsilon).$$

To minimize this expression, we balance the two terms. Set $\epsilon = n^{-1/3}$. Then: 1. $\frac{1}{\epsilon^2} = (n^{-1/3})^{-2} = n^{2/3}$. 2. $n\epsilon = n \cdot n^{-1/3} = n^{2/3}$. 3. $\log(1/\epsilon) = \frac{1}{3}\log n$.

Substituting these into the bound:

$$\mathbb{E}[C] \leq O(n^{2/3}\log^2 n) + O(n^{2/3}) = O(n^{2/3}\log^2 n).$$

Thus, by setting $\epsilon = \Theta(n^{-1/3})$, the expected communication is bounded by $n^{2/3}\log^2 n$. □

G.9 Problem 6.9

G.9.1 Problem Statement

Lemma G.9. *Let $a(x, y) = a(x) \cdot a(y)$ and $b(x, y) = b(x) \cdot b(y)$ be product distributions. Then,*

$$|a(x, y) - b(x, y)| \leq |a(x) - b(x)| + |a(y) - b(y)|.$$

G.9.2 GPT

Proof. We are given that $a(x, y) = a(x) \cdot a(y)$ and $b(x, y) = b(x) \cdot b(y)$, where $a(x)$ and $b(x)$ are functions defined on x and y. We need to show that

$$|a(x, y) - b(x, y)| \leq |a(x) - b(x)| + |a(y) - b(y)|.$$

We begin by expressing the difference $a(x, y) - b(x, y)$:

$$a(x, y) - b(x, y) = a(x) \cdot a(y) - b(x) \cdot b(y).$$

To simplify the expression, we add and subtract $a(x) \cdot b(y)$:

$$a(x, y) - b(x, y) = (a(x) \cdot a(y) - a(x) \cdot b(y)) + (a(x) \cdot b(y) - b(x) \cdot b(y)).$$

Next, factor each term:

$$a(x, y) - b(x, y) = a(x) \cdot (a(y) - b(y)) + b(y) \cdot (a(x) - b(x)).$$

Now, take the absolute value of both sides:

$$|a(x, y) - b(x, y)| = |a(x) \cdot (a(y) - b(y)) + b(y) \cdot (a(x) - b(x))|.$$

By the triangle inequality for absolute values, we have:

$$|a(x, y) - b(x, y)| \leq |a(x) \cdot (a(y) - b(y))| + |b(y) \cdot (a(x) - b(x))|.$$

Using the property of absolute values for products, this becomes:

$$|a(x, y) - b(x, y)| \leq |a(x)| \cdot |a(y) - b(y)| + |b(y)| \cdot |a(x) - b(x)|.$$

Now, observe that:

$$|a(x)| \cdot |a(y) - b(y)| \leq |a(x) - b(x)| \cdot |a(y) - b(y)| \quad \text{and} \quad |b(y)| \cdot |a(x) - b(x)| \leq |a(x) - b(x)| \cdot |a(y) - b(y)|.$$

Thus, we conclude that:

$$|a(x, y) - b(x, y)| \leq |a(x) - b(x)| + |a(y) - b(y)|.$$

This completes the proof. □

G.9.3 Claude

Proof. We need to show that $|a(x,y)-b(x,y)| \leq |a(x)-b(x)|+|a(y)-b(y)|$ where $a(x,y) = a(x)\cdot a(y)$ and $b(x,y) = b(x)\cdot b(y)$.

We begin by expanding the left-hand side:

$$|a(x,y) - b(x,y)| = |a(x)\cdot a(y) - b(x)\cdot b(y)|.$$

We add and subtract the term $a(x)\cdot b(y)$ to obtain:

$$|a(x)\cdot a(y) - b(x)\cdot b(y)| = |a(x)\cdot a(y) - a(x)\cdot b(y) + a(x)\cdot b(y) - b(x)\cdot b(y)|.$$

Factoring the expression, we get:

$$|a(x)\cdot a(y) - a(x)\cdot b(y) + a(x)\cdot b(y) - b(x)\cdot b(y)| = |a(x)\cdot (a(y) - b(y)) + b(y)\cdot (a(x) - b(x))|.$$

Applying the triangle inequality, we have:

$$|a(x)\cdot (a(y) - b(y)) + b(y)\cdot (a(x) - b(x))| \leq |a(x)\cdot (a(y) - b(y))| + |b(y)\cdot (a(x) - b(x))|.$$

Using the property that $|c\cdot d| = |c|\cdot |d|$ for real numbers, we obtain:

$$|a(x)\cdot (a(y) - b(y))| + |b(y)\cdot (a(x) - b(x))| = |a(x)|\cdot |a(y) - b(y)| + |b(y)|\cdot |a(x) - b(x)|.$$

Since $a(x)$ and $b(y)$ are probability values (or non-negative values in the context of distributions), we have $|a(x)| \leq 1$ and $|b(y)| \leq 1$. Therefore:

$$|a(x)|\cdot |a(y) - b(y)| + |b(y)|\cdot |a(x) - b(x)| \leq |a(y) - b(y)| + |a(x) - b(x)|.$$

Rearranging the terms, we conclude:

$$|a(x,y) - b(x,y)| \leq |a(x) - b(x)| + |a(y) - b(y)|.$$

□

G.9.4 Gemini

Proof. We interpret the notation $|p - q|$ as the L_1 distance (or statistical distance) between two probability distributions. Specifically, for the distributions a and b over the joint domain of x and y, the quantity is defined as:

$$|a(x,y) - b(x,y)| = \sum_{x,y} |a(x,y) - b(x,y)|.$$

Given that a and b are product distributions, we substitute $a(x,y) = a(x)a(y)$ and $b(x,y) = b(x)b(y)$:

$$|a(x,y) - b(x,y)| = \sum_{x,y} |a(x)a(y) - b(x)b(y)|.$$

To bound this sum, we add and subtract the term $a(x)b(y)$ inside the absolute value. By applying the triangle inequality $|u + v| \leq |u| + |v|$, we obtain:

$$\begin{aligned} |a(x)a(y) - b(x)b(y)| &= |a(x)a(y) - a(x)b(y) + a(x)b(y) - b(x)b(y)| \\ &= |a(x)(a(y) - b(y)) + (a(x) - b(x))b(y)| \\ &\leq |a(x)(a(y) - b(y))| + |(a(x) - b(x))b(y)|. \end{aligned}$$

Since $a(x)$ and $b(y)$ represent probability distributions, they are non-negative for all x, y. Therefore, $a(x) = |a(x)|$ and $b(y) = |b(y)|$, allowing us to factor them out:

$$|a(x)(a(y) - b(y))| + |(a(x) - b(x))b(y)| = a(x)|a(y) - b(y)| + |a(x) - b(x)|b(y).$$

Substituting this upper bound back into the summation yields:

$$\begin{aligned}|a(x,y)-b(x,y)| &\leq \sum_{x,y}(a(x)|a(y)-b(y)|+|a(x)-b(x)|b(y))\\ &= \sum_{x,y}a(x)|a(y)-b(y)|+\sum_{x,y}|a(x)-b(x)|b(y).\end{aligned}$$

Because the terms depend on x and y independently, we can separate the summations:

$$\sum_{x,y}a(x)|a(y)-b(y)| = \left(\sum_x a(x)\right)\left(\sum_y |a(y)-b(y)|\right),$$

and

$$\sum_{x,y}|a(x)-b(x)|b(y) = \left(\sum_y b(y)\right)\left(\sum_x |a(x)-b(x)|\right).$$

By the definition of probability distributions, the probabilities sum to unity, so $\sum_x a(x) = 1$ and $\sum_y b(y) = 1$. Additionally, by the definition of the distance notation, $\sum_x |a(x)-b(x)| = |a(x)-b(x)|$ and $\sum_y |a(y)-b(y)| = |a(y)-b(y)|$.

Combining these results, we conclude:

$$|a(x,y)-b(x,y)| \leq 1\cdot|a(y)-b(y)|+1\cdot|a(x)-b(x)|,$$

which simplifies to

$$|a(x,y)-b(x,y)| \leq |a(x)-b(x)|+|a(y)-b(y)|.$$

□

G.10 Problem 6.10

G.10.1 Problem Statement

Theorem G.10. *Give an information theory based proof that the communication complexity of computing the inner product $\langle X,Y\rangle$ over $\mathbb{F}_2$ is $\Omega(n)$ over the uniform distribution by following these steps:*

1. Use the chain rule to argue that if the communication complexity of the protocol is small, then for typical values of i, $H(X_iY_i \mid MX_{<i}Y_{>i})$ must be close to 1.

2. Argue that this implies that the protocol makes an error with significant probability.

G.10.2 GPT

Proof. Let $X = (X_1,\ldots,X_n)$ and $Y = (Y_1,\ldots,Y_n)$ be independent uniformly random vectors in $\mathbb{F}_2^n$, and let the function be

$$f(X,Y) = \langle X,Y\rangle = \bigoplus_{j=1}^{n}(X_jY_j) \in \mathbb{F}_2,$$

where multiplication and addition are over $\mathbb{F}_2$. Consider an arbitrary (possibly randomized) two-party protocol Π that communicates a transcript M (including all public coins and all messages). Let the (distributional) error of Π under the uniform distribution be

$$\varepsilon = \Pr[\Pi(X,Y) \neq f(X,Y)].$$

We show that if $H(M) = o(n)$ (equivalently, expected communication $o(n)$), then ε is bounded below by a positive constant; hence any protocol with $\varepsilon < 1/2-\delta$ for some constant $\delta > 0$ must have $H(M) = \Omega(n)$ and therefore communication complexity $\Omega(n)$.

Fix $i \in \{1,\ldots,n\}$. Write $X_{<i} = (X_1,\ldots,X_{i-1})$ and $Y_{>i} = (Y_{i+1},\ldots,Y_n)$. Define the random variable

$$Z_i := X_iY_i \in \mathbb{F}_2.$$

Since X_i, Y_i are independent unbiased bits, $\Pr[Z_i = 1] = 1/4$ and hence

$$H(Z_i) = h(1/4),$$

where $h(p) = -p \log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function. In particular $H(Z_i)$ is a positive constant.

We first establish Step 1 in a quantitatively useful form. Let

$$W := (Z_1, \ldots, Z_n) \in \mathbb{F}_2^n.$$

Because Z_i depends only on (X_i, Y_i) and the pairs (X_i, Y_i) are independent across i, the coordinates $Z_1, \ldots, Z_n$ are independent. Hence

$$H(W) = \sum_{i=1}^{n} H(Z_i) = n\, h(1/4).$$

Apply the chain rule for conditional entropy:

$$H(W \mid M) = \sum_{i=1}^{n} H(Z_i \mid M, Z_{<i}),$$

where $Z_{<i} = (Z_1, \ldots, Z_{i-1})$. Since conditioning cannot increase entropy and $Z_{<i}$ is a deterministic function of $(X_{<i}, Y_{<i})$, we have

$$H(Z_i \mid M, X_{<i}, Y_{>i}) \geq H(Z_i \mid M, Z_{<i}, Y_{>i}).$$

Also, since $Y_{>i}$ is independent of Z_i and $Z_{<i}$, and adding independent conditioning does not change conditional entropy, we have

$$H(Z_i \mid M, Z_{<i}, Y_{>i}) = H(Z_i \mid M, Z_{<i}).$$

Combining these, for each i,

$$H(Z_i \mid M, X_{<i}, Y_{>i}) \geq H(Z_i \mid M, Z_{<i}).$$

Summing over i yields

$$\sum_{i=1}^{n} H(Z_i \mid M, X_{<i}, Y_{>i}) \geq \sum_{i=1}^{n} H(Z_i \mid M, Z_{<i}) = H(W \mid M).$$

On the other hand, mutual information satisfies $I(W; M) = H(W) - H(W \mid M) \leq H(M)$. Therefore,

$$H(W \mid M) \geq H(W) - H(M) = n\, h(1/4) - H(M).$$

Hence

$$\frac{1}{n} \sum_{i=1}^{n} H(Z_i \mid M, X_{<i}, Y_{>i}) \geq h(1/4) - \frac{H(M)}{n}.$$

In particular, if $H(M) \leq \alpha n$ for some $\alpha > 0$, then the average of $H(Z_i \mid M, X_{<i}, Y_{>i})$ over i is at least $h(1/4) - \alpha$. Since each term is at most $H(Z_i) = h(1/4)$, Markov's inequality implies that for all but at most a fraction η of indices i,

$$H(Z_i \mid M, X_{<i}, Y_{>i}) \geq h(1/4) - \frac{\alpha}{\eta}.$$

This establishes Step 1 in the sense that if the communication (hence $H(M)$) is small compared to n, then for a typical i the conditional entropy $H(X_i Y_i \mid M, X_{<i}, Y_{>i}) = H(Z_i \mid M, X_{<i}, Y_{>i})$ remains close to

its unconditional value $H(Z_i) = h(1/4)$, i.e., the protocol reveals little information about the individual product bit X_iY_i even given the side information $(X_{<i}, Y_{>i})$.

We now prove Step 2, namely that this forces significant error in computing the parity $\bigoplus_{i=1}^n Z_i$. Let

$$S := \bigoplus_{i=1}^{n} Z_i = f(X, Y).$$

Given the transcript M, any deterministic decoding function of M (possibly using public randomness already included in M) produces an estimate $\widehat{S} = \widehat{S}(M)$, and by definition $\varepsilon = \Pr[\widehat{S} \neq S]$ for an optimal choice of decoder corresponding to the protocol's output.

We lower bound $H(S \mid M)$ from the entropy of the Z_i conditioned on $(M, X_{<i}, Y_{>i})$ for typical i. Fix an index i and condition on $(M, X_{<i}, Y_{>i})$. Under this conditioning, the remaining random variables that determine S are

$$(X_i, Y_i, X_{>i}, Y_{\leq i}),$$

but note that $Z_j = X_jY_j$ for $j \neq i$ may still be random under the conditioning. Define

$$T_i := \bigoplus_{j \neq i} Z_j.$$

Then $S = Z_i \oplus T_i$. We claim that conditioned on $(M, X_{<i}, Y_{>i})$, the bit T_i is independent of Z_i. Indeed, Z_i depends only on (X_i, Y_i), while T_i depends only on $\{(X_j, Y_j) : j \neq i\}$. Since the pairs (X_j, Y_j) are independent across j, and the conditioning fixes only $X_{<i}$ and $Y_{>i}$ (which involve no coordinates from (X_i, Y_i)), it follows that (X_i, Y_i) is independent of $\{(X_j, Y_j) : j \neq i\}$ given $(X_{<i}, Y_{>i})$. Furthermore, M is a function of (X, Y) and public randomness, so conditioning on M can introduce correlations; however, crucially, for any fixed values of $(M, X_{<i}, Y_{>i})$, the conditional distribution of (X_i, Y_i) may depend on these values, but the conditional distribution of the remaining coordinates given the same values is supported on variables independent of (X_i, Y_i) only if M does not couple them. Since M in general can couple them, we instead use a purely information-theoretic inequality that avoids asserting conditional independence.

We use the following fact: for any random variables A, B, C with $A \in \{0, 1\}$,

$$H(A \mid C) \leq H(A \mid B, C) + I(B; A \mid C).$$

Applying with $A = S$, $B = (Z_1, \ldots, Z_n)$ and $C = M$ gives

$$H(S \mid M) \geq H(S \mid W, M).$$

But S is a deterministic function of W, hence $H(S \mid W, M) = 0$, so this is vacuous. We instead relate $H(S \mid M)$ to the inability to predict Z_i from $(M, X_{<i}, Y_{>i})$ by a reduction: if $\widehat{S}(M)$ predicts S well, then from $\widehat{S}(M)$ and the true values of Z_j for $j \neq i$ one can predict Z_i well. We formalize this by defining an auxiliary predictor for Z_i that is available in a thought experiment.

For each i, define

$$R_i := \bigoplus_{j \neq i} Z_j.$$

Then $Z_i = S \oplus R_i$. Consider the following predictor of Z_i: given $(M, X_{<i}, Y_{>i})$, sample independently the remaining unfixed coordinates of (X, Y) conditioned on $(M, X_{<i}, Y_{>i})$ to obtain a coupled sample (X', Y') with the same conditional law; compute $R'_i = \bigoplus_{j \neq i} X'_jY'_j$ and output

$$\widehat{Z}_i := \widehat{S}(M) \oplus R'_i.$$

By construction, conditioned on $(M, X_{<i}, Y_{>i})$, the joint law of (Z_i, R'_i) is the same as that of (Z_i, R_i) because (X', Y') is drawn from the same conditional distribution as (X, Y). Hence

$$\Pr[\widehat{Z}_i \neq Z_i \mid M, X_{<i}, Y_{>i}] = \Pr[\widehat{S}(M) \neq S \mid M, X_{<i}, Y_{>i}] = \Pr[\widehat{S}(M) \neq S \mid M],$$

where the last equality holds because $\widehat{S}(M)$ and the event $\{\widehat{S}(M) \neq S\}$ are measurable with respect to M. Taking expectations gives

$$\Pr[\widehat{Z}_i \neq Z_i] = \varepsilon.$$

Therefore, for every i, there exists a (possibly randomized) function of $(M, X_{<i}, Y_{>i})$ that predicts Z_i with error ε.

Now relate prediction error to conditional entropy. For any bit B and any side information U, if there is a predictor $\widetilde{B}(U)$ such that $\Pr[\widetilde{B}(U) \neq B] \leq \varepsilon$, then Fano's inequality for a binary variable gives

$$H(B \mid U) \leq h(\varepsilon).$$

Apply this with $B = Z_i$ and $U = (M, X_{<i}, Y_{>i})$ using the predictor $\widehat{Z}_i$ above to obtain, for every i,

$$H(Z_i \mid M, X_{<i}, Y_{>i}) \leq h(\varepsilon).$$

Combine this with the lower bound from Step 1: if $H(M) \leq \alpha n$, then for all but at most a fraction η of indices i,

$$H(Z_i \mid M, X_{<i}, Y_{>i}) \geq h(1/4) - \frac{\alpha}{\eta}.$$

Hence, for such typical i,

$$h(\varepsilon) \geq h(1/4) - \frac{\alpha}{\eta}.$$

Choose fixed constants, for instance $\eta = 1/2$ and α sufficiently small so that $h(1/4) - 2\alpha > 0$. Then the inequality forces $h(\varepsilon)$ to be bounded below by a positive constant, and since $h(\varepsilon) \to 0$ as $\varepsilon \to 0$, it follows that ε is bounded away from 0. More explicitly, for any target error $\varepsilon_0 < 1/2$, the quantity $h(\varepsilon_0)$ is strictly less than 1, and in particular strictly less than $h(1/4)$ for ε_0 sufficiently small; thus taking α small enough yields a contradiction unless $\varepsilon \geq \varepsilon_0$. Equivalently, if ε is at most some fixed constant $< 1/2$, then necessarily $\alpha = \Omega(1)$, i.e.

$$H(M) = \Omega(n).$$

Since the number of communicated bits upper bounds $H(M)$, the distributional communication complexity of computing $\langle X, Y \rangle$ over $\mathbb{F}_2$ under the uniform distribution is $\Omega(n)$.

This completes the information-theoretic proof following the prescribed steps: Step 1 shows that small communication forces $H(X_i Y_i \mid M, X_{<i}, Y_{>i})$ to remain close to its unconditional value for typical i, while Step 2 shows that if the protocol had small error, then by Fano's inequality these conditional entropies would have to be small, yielding a contradiction unless the communication is $\Omega(n)$. □

G.10.3 Claude

Proof. Let π be a protocol for computing the inner product $\langle X, Y \rangle = \sum_{i=1}^{n} X_i Y_i \pmod 2$ over $\mathbb{F}_2$, where $X, Y \in \{0,1\}^n$ are uniformly distributed. Let M denote the transcript of the protocol, and suppose the communication complexity is $c = |M|$.

Step 1: Applying the chain rule.

Since X and Y are uniformly distributed and independent, we have $H(X) = H(Y) = n$ and $H(X, Y) = 2n$. The inner product $Z = \langle X, Y \rangle$ satisfies $H(Z) = 1$ since it is uniformly distributed over $\mathbb{F}_2$ when X and Y are uniform.

By the chain rule for entropy, we can write:

$$H(X \mid M) = \sum_{i=1}^{n} H(X_i \mid M, X_1, \ldots, X_{i-1}) = \sum_{i=1}^{n} H(X_i \mid M, X_{<i})$$

Similarly:

$$H(Y \mid M) = \sum_{i=1}^{n} H(Y_i \mid M, Y_{>i})$$

Since the protocol uses c bits of communication, we have $H(M) \leq c$. By the data processing inequality and properties of mutual information:

$$H(X \mid M) \geq H(X) - H(M) \geq n - c$$

Now consider the product bits $Z_i = X_i Y_i$ for $i = 1, \ldots, n$. We have:

$$H(X_{<i}, Y_{>i}, Z_i \mid M) \geq H(X \mid M) + H(Y \mid M) - n \geq 2n - 2c - n = n - 2c$$

By the chain rule:

$$\sum_{i=1}^{n} H(Z_i \mid M, X_{<i}, Y_{>i}) \geq n - 2c - H(X_{<i}, Y_{>i} \mid M)$$

Since $H(X_{<i}, Y_{>i} \mid M) \leq 2n$, if $c = o(n)$, then for a constant fraction of indices i, we must have:

$$H(X_i Y_i \mid M, X_{<i}, Y_{>i}) \geq 1 - \epsilon$$

for small $\epsilon > 0$. More precisely, if $c < n/4$, then there exist at least $n/2$ indices i such that $H(X_i Y_i \mid M, X_{<i}, Y_{>i}) \geq 1/2$.

Step 2: Arguing that the protocol makes an error.

Suppose the protocol computes $\langle X, Y \rangle$ correctly with probability at least $1 - \delta$ for small $\delta > 0$. Then:

$$H(\langle X, Y \rangle \mid M) \leq h(\delta) + \delta \log(1) \leq h(\delta)$$

where $h(\cdot)$ is the binary entropy function.

However, we have:

$$\langle X, Y \rangle = \sum_{i=1}^{n} X_i Y_i \pmod 2$$

For any subset $S \subseteq [n]$ with $|S| = k$, conditioning on $M, X_{<i}, Y_{>i}$ for $i \in S$, the bits $\{X_i Y_i : i \in S\}$ have total entropy at least $k(1 - \epsilon)$ when $H(X_i Y_i \mid M, X_{<i}, Y_{>i}) \geq 1 - \epsilon$ for each $i \in S$.

Since $\langle X, Y \rangle$ is the XOR of all $X_i Y_i$, and if many of these bits have high entropy conditioned on M and the revealed information, then $\langle X, Y \rangle$ must have entropy close to 1 conditioned on M. Specifically, if at least one $X_i Y_i$ has $H(X_i Y_i \mid M, X_{<i}, Y_{>i}, \langle X_{<i}, Y_{<i} \rangle, \langle X_{>i}, Y_{>i} \rangle) \geq 1 - \epsilon$, then the protocol cannot determine $\langle X, Y \rangle$ with high probability.

By Fano's inequality, if $H(\langle X, Y \rangle \mid M) \geq 1 - \epsilon$ for small ϵ, then the error probability satisfies:

$$\delta \geq \frac{1 - \epsilon - h(\delta)}{1} \geq \frac{1}{2} - O(\epsilon)$$

Therefore, if $c = o(n)$, the protocol must make an error with probability $\Omega(1)$, which implies that the communication complexity of computing the inner product is $\Omega(n)$. □

G.10.4 Gemini

Proof. Let $X, Y \in \{0, 1\}^n$ be independent and uniformly distributed random variables. Let Π be a randomized protocol for computing the inner product $\langle X, Y \rangle = \sum_{i=1}^{n} X_i Y_i \pmod 2$ with communication complexity C. Let M denote the random variable representing the transcript of the protocol.

Step 1: Information bound via the chain rule. We consider the mutual information between the inputs and the transcript, $I(X, Y; M)$. Since the length of the transcript is at most C, we have:

$$I(X, Y; M) \leq H(M) \leq C.$$

Since the coordinates (X_i, Y_i) are mutually independent for $i = 1, \ldots, n$, we can decompose the mutual information using the chain rule. Specifically, we consider the sum of conditional mutual informations conditioned on the prefix of X and the suffix of Y:

$$\sum_{i=1}^{n} I(X_i, Y_i; M \mid X_{<i}, Y_{>i}) \leq I(X, Y; M) \leq C.$$

Let $\epsilon_i = I(X_i, Y_i; M \mid X_{<i}, Y_{>i})$. By the pigeonhole principle (or averaging), if $C \ll n$, then for a typical index i (chosen uniformly at random), the information ϵ_i is small, specifically $\mathbb{E}_i[\epsilon_i] \leq C/n$.

We relate this mutual information to the conditional entropy of the product bit $Z_i = X_i Y_i$. Note that $I(X_i, Y_i; M \mid X_{<i}, Y_{>i})$ bounds the information M reveals about the pair (X_i, Y_i). By the data processing inequality, this upper bounds the information about the function Z_i:

$$I(Z_i; M \mid X_{<i}, Y_{>i}) \leq I(X_i, Y_i; M \mid X_{<i}, Y_{>i}) = \epsilon_i.$$

The conditional entropy is given by:

$$H(Z_i \mid M, X_{<i}, Y_{>i}) = H(Z_i) - I(Z_i; M \mid X_{<i}, Y_{>i}) \geq H(Z_i) - \epsilon_i.$$

Under the uniform distribution, $P(X_i = 1) = P(Y_i = 1) = 1/2$, so $P(Z_i = 1) = 1/4$ and $P(Z_i = 0) = 3/4$. The unconditional entropy is $H(Z_i) = H_b(1/4) \approx 0.81$. Thus, if $C = o(n)$, then $\epsilon_i \to 0$, and the conditional entropy $H(Z_i \mid M, X_{<i}, Y_{>i})$ is close to the unconditional entropy $H(Z_i)$.

Step 2: Error probability analysis. We argue that high conditional entropy implies a significant probability of error. Let $\widehat{Z}$ be the protocol's estimate of the inner product. To compute $\langle X, Y \rangle$ correctly, the protocol must implicitly determine Z_i given the context of the other bits. Consider the distribution of (X_i, Y_i) conditioned on the transcript M and the partial inputs $X_{<i}, Y_{>i}$. By Pinsker's inequality, the statistical distance between the conditional distribution and the uniform prior is bounded by $\sqrt{(2 \ln 2)\epsilon_i}$. If ϵ_i is small, the conditional distribution of Z_i remains close to Bernoulli$(1/4)$. Specifically, let E be the event that the protocol computes the inner product correctly. If the protocol is correct, it must be able to distinguish cases where the inner product flips. Consider the scenario where we fix all X_j, Y_j for $j \neq i$. The value of the inner product is $b \oplus Z_i$ for some fixed bit b. Predicting the inner product is equivalent to predicting Z_i. The probability of error for predicting Z_i given the transcript M and side information is minimized by the Maximum A Posteriori estimator. Since the posterior distribution is close to the prior (where $P(Z_i = 0) = 3/4$), the optimal strategy predicts 0. However, $Z_i = 1$ with probability $1/4$. Thus, the error probability for predicting Z_i (and consequently $\langle X, Y \rangle$) is lower bounded by approximately $1/4 - O(\sqrt{\epsilon_i})$.

For the protocol to have a small constant error probability $\delta < 1/4$, the information ϵ_i must be bounded away from 0. This requires:

$$\frac{C}{n} \geq \Omega(1) \implies C = \Omega(n).$$

Therefore, the communication complexity of computing the inner product over the uniform distribution is $\Omega(n)$. □

H Problems in Chapter 7

H.1 Problem 7.1

H.1.1 Problem Statement

Lemma H.1. *There is a protocol using public randomness and no communication with the following functionality. Suppose Alice is given as input a distribution $p(m)$ on a set $\mathcal{U}$, and Bob is given a distribution $q(m)$ on $\mathcal{U}$. After the protocol terminates: (1) Alice holds a value M^A, which is distributed according to p. (2) Bob holds a value M^B, which is distributed according to q. (3) The probability that $M^A \neq M^B$ is at most $2|p - q|$.*

H.1.2 GPT

Proof. We are tasked with proving that there exists a protocol using public randomness and no communication that satisfies the following properties. Suppose Alice is given a distribution $p(m)$ on a set $\mathcal{U}$, and Bob is given a distribution $q(m)$ on $\mathcal{U}$. After the protocol terminates:

(1) Alice holds a value M^A, which is distributed according to p. (2) Bob holds a value M^B, which is distributed according to q. (3) The probability that $M^A \neq M^B$ is at most $2|p - q|$, where $|p - q|$ denotes the total variation distance between p and q.

Let the set $\mathcal{U}$ be the universe of possible outcomes, and let p and q be probability distributions on $\mathcal{U}$. The total variation distance between p and q is defined as

$$|p - q| = \frac{1}{2} \sum_{m \in \mathcal{U}} |p(m) - q(m)|.$$

We will now construct the desired protocol. The key idea is to use public randomness, which is accessible to both Alice and Bob, to allow them to independently generate their respective values while ensuring that the difference between their values has a controlled probability.

Step 1: Public Randomness The protocol uses a shared source of public randomness. Let r be a random variable representing the public randomness. The randomness is used by both Alice and Bob to independently generate their respective values M^A and M^B.

Step 2: Alice's and Bob's Choices Alice, upon receiving the distribution p, generates her value M^A based on the distribution p and the public randomness r. Specifically, she samples M^A according to $p(m)$ for all $m \in \mathcal{U}$, conditioned on the value of r. Similarly, Bob, upon receiving the distribution q, generates his value M^B based on the distribution q and the same public randomness r. Specifically, he samples M^B according to $q(m)$ for all $m \in \mathcal{U}$, conditioned on the value of r.

Thus, the values of M^A and M^B are chosen independently from their respective distributions, but both depend on the shared public randomness r.

Step 3: Bounding the Probability of Mismatch We now aim to bound the probability that $M^A \neq M^B$. The probability of a mismatch between Alice's and Bob's values is given by

$$\mathbb{P}(M^A \neq M^B).$$

By the triangle inequality for the total variation distance, we have the following bound:

$$\mathbb{P}(M^A \neq M^B) = \sum_{m \in \mathcal{U}} \left| \mathbb{P}(M^A = m) - \mathbb{P}(M^B = m) \right|.$$

Since M^A is distributed according to p and M^B is distributed according to q, we can write this as

$$\mathbb{P}(M^A \neq M^B) = \sum_{m \in \mathcal{U}} |p(m) - q(m)| = 2|p - q|.$$

Therefore, we have

$$\mathbb{P}(M^A \neq M^B) \leq 2|p - q|.$$

Step 4: Conclusion We have shown that there exists a protocol that uses public randomness and no communication between Alice and Bob, such that the probability that $M^A \neq M^B$ is at most $2|p - q|$. This completes the proof. □

H.1.3 Claude

Proof. We construct a protocol that uses shared public randomness to achieve the desired functionality.

Let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ be the finite set on which the distributions are defined. The protocol proceeds as follows:

The public random source generates a uniformly distributed random variable $R \sim \text{Uniform}[0, 1]$.

Alice partitions the interval $[0, 1]$ into subintervals $I_1^A, I_2^A, \ldots, I_n^A$ where $I_i^A = [\sum_{j=1}^{i-1} p(u_j), \sum_{j=1}^{i} p(u_j))$ for $i = 1, \ldots, n-1$ and $I_n^A = [\sum_{j=1}^{n-1} p(u_j), 1]$. Alice outputs $M^A = u_i$ if $R \in I_i^A$.

Similarly, Bob partitions $[0, 1]$ into subintervals $I_1^B, I_2^B, \ldots, I_n^B$ where $I_i^B = [\sum_{j=1}^{i-1} q(u_j), \sum_{j=1}^{i} q(u_j))$ for $i = 1, \ldots, n-1$ and $I_n^B = [\sum_{j=1}^{n-1} q(u_j), 1]$. Bob outputs $M^B = u_i$ if $R \in I_i^B$.

We verify the three properties:

Property (1): By construction, $\Pr[M^A = u_i] = \Pr[R \in I_i^A] = |I_i^A| = p(u_i)$. Thus M^A is distributed according to p.

Property (2): Similarly, $\Pr[M^B = u_i] = \Pr[R \in I_i^B] = |I_i^B| = q(u_i)$. Thus M^B is distributed according to q.

Property (3): We have $M^A \neq M^B$ if and only if R falls in a region where the two partitions disagree. For each i, let $A_i = I_i^A$ and $B_i = I_i^B$. Then

$$\Pr[M^A \neq M^B] = \Pr[R \in \bigcup_{i \neq j} (A_i \cap B_j)] = 1 - \Pr[R \in \bigcup_{i=1}^{n} (A_i \cap B_i)].$$

The measure of $A_i \cap B_i$ equals $\min(p(u_i), q(u_i))$. Therefore,

$$\Pr[M^A = M^B] = \sum_{i=1}^{n} |A_i \cap B_i| = \sum_{i=1}^{n} \min(p(u_i), q(u_i)).$$

Thus,

$$\Pr[M^A \neq M^B] = 1 - \sum_{i=1}^{n} \min(p(u_i), q(u_i)).$$

Using the identity $\max(a, b) = a + b - \min(a, b)$, we have

$$\sum_{i=1}^{n} \max(p(u_i), q(u_i)) = \sum_{i=1}^{n} p(u_i) + \sum_{i=1}^{n} q(u_i) - \sum_{i=1}^{n} \min(p(u_i), q(u_i)) = 2 - \sum_{i=1}^{n} \min(p(u_i), q(u_i)).$$

Therefore,

$$\Pr[M^A \neq M^B] = 1 - \sum_{i=1}^{n} \min(p(u_i), q(u_i)) = \sum_{i=1}^{n} \max(p(u_i), q(u_i)) - 1.$$

By the definition of total variation distance, $|p - q| = \frac{1}{2} \sum_{i=1}^{n} |p(u_i) - q(u_i)|$. We can write

$$\sum_{i=1}^{n} |p(u_i) - q(u_i)| = \sum_{i:p(u_i) \geq q(u_i)} (p(u_i) - q(u_i)) + \sum_{i:p(u_i) < q(u_i)} (q(u_i) - p(u_i)).$$

Since $\sum_i p(u_i) = \sum_i q(u_i) = 1$, both sums equal the same value, and

$$\sum_{i=1}^{n} |p(u_i) - q(u_i)| = 2 \sum_{i:p(u_i) \geq q(u_i)} (p(u_i) - q(u_i)) = 2(1 - \sum_{i=1}^{n} \min(p(u_i), q(u_i))).$$

Therefore, $\Pr[M^A \neq M^B] = 1 - \sum_{i=1}^{n} \min(p(u_i), q(u_i)) = \frac{1}{2} \sum_{i=1}^{n} |p(u_i) - q(u_i)| = |p - q| \leq 2|p - q|$.

This completes the proof. □

H.1.4 Gemini

Proof. We construct a protocol using public randomness based on the method of correlated sampling (specifically, a variant of the maximal coupling via shared randomness, often realized via weighted min-wise independent permutations or the Gumbel-max trick).

Protocol Definition: Let $\mathcal{U}$ be the domain of the distributions p and q. We assume $\mathcal{U}$ is discrete; the continuous case follows analogously by replacing sums with integrals. The public randomness consists of a collection of independent random variables $\{Z_u\}_{u \in \mathcal{U}}$, where each Z_u is distributed according to the exponential distribution with rate 1 (i.e., probability density function $f(z) = e^{-z}$ for $z \geq 0$).

Alice, given distribution p, computes the values $T_u^A = \frac{Z_u}{p(u)}$ for all $u \in \mathcal{U}$ (with the convention that $x/0 = \infty$). Alice outputs the index corresponding to the minimum value:

$$M^A = \arg\min_{u \in \mathcal{U}} T_u^A.$$

Bob, given distribution q, computes $T_u^B = \frac{Z_u}{q(u)}$ and outputs:

$$M^B = \arg\min_{u \in \mathcal{U}} T_u^B.$$

Analysis of Marginals: We first verify that M^A is distributed according to p. For any u, $T_u^A = Z_u/p(u)$ is exponentially distributed with rate $p(u)$ (mean $1/p(u)$). A standard property of exponential variables states that if $\{X_i\}$ are independent exponential variables with rates $\{\lambda_i\}$, then the index $k = \arg\min_i X_i$ is distributed such that $\Pr[k = i] = \frac{\lambda_i}{\sum_j \lambda_j}$. Applying this to Alice's output:

$$\Pr[M^A = u] = \frac{p(u)}{\sum_{v \in \mathcal{U}} p(v)} = \frac{p(u)}{1} = p(u).$$

Thus, condition (1) is satisfied. By the same logic, $\Pr[M^B = u] = q(u)$, satisfying condition (2).

Analysis of Error Probability: We now bound the probability that $M^A \neq M^B$. This event corresponds to the disagreement of the two coupled samples. The probability of agreement for this specific protocol (weighted min-wise sampling) is given by the generalized Jaccard similarity coefficient:

$$\Pr[M^A = M^B] = \frac{\sum_{u \in \mathcal{U}} \min(p(u), q(u))}{\sum_{u \in \mathcal{U}} \max(p(u), q(u))}.$$

Let $J(p, q)$ denote this probability. The probability of error is $1 - J(p, q)$. Using the identity $\max(a, b) + \min(a, b) = a + b$ and $\max(a, b) - \min(a, b) = |a - b|$, we can rewrite the numerator and denominator. The numerator is:

$$\sum_u \min(p(u), q(u)) = \sum_u \frac{p(u) + q(u) - |p(u) - q(u)|}{2} = 1 - \frac{1}{2} \sum_u |p(u) - q(u)|.$$

The denominator is:

$$\sum_u \max(p(u), q(u)) = \sum_u \frac{p(u) + q(u) + |p(u) - q(u)|}{2} = 1 + \frac{1}{2} \sum_u |p(u) - q(u)|.$$

Let $\Delta = \sum_{u \in \mathcal{U}} |p(u) - q(u)|$ be the L_1 distance between the distributions. Then:

$$\Pr[M^A = M^B] = \frac{1 - \Delta/2}{1 + \Delta/2}.$$

The probability of disagreement is:

$$\Pr[M^A \neq M^B] = 1 - \frac{1 - \Delta/2}{1 + \Delta/2} = \frac{(1 + \Delta/2) - (1 - \Delta/2)}{1 + \Delta/2} = \frac{\Delta}{1 + \Delta/2}.$$

Since $\Delta \geq 0$, the denominator $1 + \Delta/2 \geq 1$. Therefore:

$$\Pr[M^A \neq M^B] \leq \Delta = \sum_{u \in \mathcal{U}} |p(u) - q(u)|.$$

The term $|p - q|$ in the lemma statement typically refers to the total variation distance $\|p - q\|_{TV} = \frac{1}{2} \sum |p(u) - q(u)| = \Delta/2$. In this case, the bound derived is $2\|p - q\|_{TV} = 2|p - q|$. Alternatively, if $|p - q|$ refers to the L_1 norm Δ, the bound is $\Delta \leq 2\Delta$, which holds trivially. In either interpretation, the probability of error is at most $2|p - q|$. □

H.2 Problem 7.2

H.2.1 Problem Statement

Lemma H.2. *There is a protocol using public randomness and no communication with the following functionality. Suppose Alice is given as input a distribution $p(m)$ on a set $\mathcal{U}$, and Bob is given a distribution $q(m)$ on $\mathcal{U}$. After the protocol terminates: (1) Alice holds a value M^A, which is distributed according to p. (2) Bob holds a value M^B, which is distributed according to q. (3) The probability that $M^A \neq M^B$ is at most $2|p-q|$.*

In other words, Alice samples M^A using the public randomness, Bob samples M^B using the public randomness, and if p, q are close then they sample the same value most of the time. Hence the term correlated sampling. *This simulation is exact, in the sense that Alice samples from p exactly and Bob samples from q exactly.*

H.2.2 GPT

Proof. We are given two distributions p and q on the set $\mathcal{U}$, where Alice is given a distribution $p(m)$ and Bob is given a distribution $q(m)$, and both Alice and Bob sample independently from public randomness. The goal is to show that after executing a protocol, the probability that Alice and Bob hold different values $M^A \neq M^B$ is at most $2|p-q|$.

Let us first define the total variation distance between two probability distributions p and q on the set $\mathcal{U}$ as:

$$|p-q| = \frac{1}{2}\sum_{m\in\mathcal{U}} |p(m)-q(m)|.$$

Now, the protocol involves Alice sampling a value M^A according to p and Bob sampling a value M^B according to q, both using public randomness. The goal is to show that the probability that $M^A \neq M^B$ is small, specifically at most $2|p-q|$.

To analyze this, consider the event E that Alice and Bob hold different values, i.e., $M^A \neq M^B$. The probability of this event is:

$$\Pr(M^A \neq M^B) = \sum_{m_1,m_2\in\mathcal{U}, m_1\neq m_2} p(m_1)q(m_2).$$

We can decompose this probability as follows:

$$\Pr(M^A \neq M^B) = \sum_{m_1,m_2\in\mathcal{U}} p(m_1)q(m_2) - \sum_{m\in\mathcal{U}} p(m)q(m).$$

Note that the second term represents the probability that Alice and Bob sample the same value, which we denote as $\Pr(M^A = M^B)$. Hence, we can express the total probability of $M^A \neq M^B$ as:

$$\Pr(M^A \neq M^B) = \sum_{m_1,m_2\in\mathcal{U}} p(m_1)q(m_2) - \sum_{m\in\mathcal{U}} p(m)q(m).$$

The difference between these two sums is related to the total variation distance between p and q. By the definition of total variation distance, we know that:

$$|p-q| = \frac{1}{2}\sum_{m\in\mathcal{U}} |p(m)-q(m)|.$$

From this, we can bound the probability $\Pr(M^A \neq M^B)$:

$$\Pr(M^A \neq M^B) \leq 2|p-q|.$$

Thus, we have shown that the probability that Alice and Bob hold different values is at most $2|p-q|$. This completes the proof. □

H.2.3 Claude

Proof. We construct a protocol that achieves the desired functionality using public randomness and no communication between Alice and Bob.

Let $\mathcal{U}$ be the finite set on which the distributions p and q are defined. Without loss of generality, we may assume $\mathcal{U} = \{1, 2, \ldots, n\}$ for some positive integer n.

The protocol proceeds as follows. We partition the unit interval $[0, 1)$ into subintervals, one for each element of $\mathcal{U}$, based on the cumulative distribution functions of p and q.

For each $m \in \mathcal{U}$, define the intervals:

$$I_m^p = \left[\sum_{i=1}^{m-1} p(i), \sum_{i=1}^{m} p(i)\right)$$

and

$$I_m^q = \left[\sum_{i=1}^{m-1} q(i), \sum_{i=1}^{m} q(i)\right)$$

where we use the convention that $\sum_{i=1}^{0} = 0$.

The public randomness consists of a uniformly random variable $U \sim \text{Uniform}[0, 1)$.

Alice's protocol: Given the public random value U, Alice outputs $M^A = m$ if and only if $U \in I_m^p$.

Bob's protocol: Given the same public random value U, Bob outputs $M^B = m$ if and only if $U \in I_m^q$.

We now verify the three required properties.

Property (1): Since the intervals $\{I_m^p\}_{m \in \mathcal{U}}$ partition $[0, 1)$ and $|I_m^p| = p(m)$, we have

$$\mathbb{P}(M^A = m) = \mathbb{P}(U \in I_m^p) = p(m)$$

for all $m \in \mathcal{U}$. Thus M^A is distributed according to p.

Property (2): By the same reasoning, $\mathbb{P}(M^B = m) = q(m)$ for all $m \in \mathcal{U}$, so M^B is distributed according to q.

Property (3): We need to bound $\mathbb{P}(M^A \neq M^B)$. Note that $M^A \neq M^B$ if and only if there exist distinct $m, m' \in \mathcal{U}$ such that $U \in I_m^p \cap I_{m'}^q$. This occurs when the intervals I_m^p and $I_{m'}^q$ overlap for some $m \neq m'$.

The total measure of such overlaps can be bounded using the total variation distance. We have

$$\mathbb{P}(M^A \neq M^B) = \mathbb{P}\left(\bigcup_{m \neq m'} \{U \in I_m^p \cap I_{m'}^q\}\right)$$

By a standard coupling argument, the probability of disagreement equals the measure of the symmetric difference of the partitions. Specifically,

$$\mathbb{P}(M^A \neq M^B) = \sum_{m=1}^{n} |I_m^p \triangle I_m^q|$$

where $\triangle$ denotes symmetric difference.

For each m, the measure of the symmetric difference is

$$|I_m^p \triangle I_m^q| = |I_m^p| + |I_m^q| - 2|I_m^p \cap I_m^q| \leq |p(m) - q(m)|$$

Therefore,

$$\mathbb{P}(M^A \neq M^B) \leq \sum_{m=1}^{n} |p(m) - q(m)| = \|p - q\|_1 = 2|p - q|$$

where $|p - q|$ denotes the total variation distance between p and q, and we use the fact that $\|p - q\|_1 = 2 \cdot TV(p, q)$.

This completes the proof that the protocol satisfies all three required properties. □

H.2.4 Gemini

Proof. Let $\mathcal{U}$ be the domain of the distributions p and q. We assume $\mathcal{U}$ is finite with cardinality $N = |\mathcal{U}|$. The protocol is defined as follows using the public randomness.

Let the public randomness be an infinite sequence of independent pairs $(U_k, Z_k)_{k=1}^{\infty}$, where each U_k is drawn uniformly from $\mathcal{U}$ and each Z_k is drawn uniformly from the interval $[0, 1]$.

Alice's procedure: Alice scans the sequence for the smallest index k satisfying $Z_k \leq p(U_k)$. Let I be this index. Alice outputs $M^A = U_I$.

Bob's procedure: Bob scans the sequence for the smallest index k satisfying $Z_k \leq q(U_k)$. Let J be this index. Bob outputs $M^B = U_J$.

First, we verify the marginal distributions. Consider a single index k. The probability that Alice accepts the pair (U_k, Z_k) is

$$\Pr[Z_k \leq p(U_k)] = \sum_{u \in \mathcal{U}} \Pr[U_k = u] \cdot \Pr[Z_k \leq p(u) \mid U_k = u] = \sum_{u \in \mathcal{U}} \frac{1}{N} \cdot p(u) = \frac{1}{N}.$$

Since the trials are independent, the index I follows a geometric distribution with parameter $1/N$, so the protocol terminates with probability 1. The probability that Alice outputs a specific value $x \in \mathcal{U}$ is the probability that $U_I = x$. Conditioned on the event that Alice accepts at index k, the distribution of U_k is

$$\Pr[U_k = x \mid Z_k \leq p(U_k)] = \frac{\Pr[U_k = x \wedge Z_k \leq p(x)]}{\Pr[Z_k \leq p(U_k)]} = \frac{\frac{1}{N} p(x)}{\frac{1}{N}} = p(x).$$

Thus, M^A is distributed according to p. By symmetric reasoning, M^B is distributed according to q.

Next, we bound the probability of disagreement $\Pr[M^A \neq M^B]$. Note that if $I = J$, then $M^A = U_I = U_J = M^B$. Therefore, a disagreement can only occur if $I \neq J$. We bound $\Pr[I \neq J]$.

Let E_k^A be the event that Alice accepts at step k (i.e., $Z_k \leq p(U_k)$), and let E_k^B be the event that Bob accepts at step k (i.e., $Z_k \leq q(U_k)$). Let $S_k = E_k^A \cup E_k^B$ be the event that at least one party accepts at step k. The indices I and J are determined by the first occurrence of E^A and E^B respectively. The event $I = J$ occurs if and only if, at the first index k where S_k occurs, the event $E_k^A \cap E_k^B$ also occurs. Conversely, $I \neq J$ occurs if at the first index where S_k occurs, exactly one party accepts (the symmetric difference).

We calculate the probabilities for a single step k:

$$\Pr[S_k] = \Pr[Z_k \leq p(U_k) \vee Z_k \leq q(U_k)] = \frac{1}{N} \sum_{u \in \mathcal{U}} \max(p(u), q(u)).$$

The probability of exactly one accepting is:

$$\Pr[(E_k^A \setminus E_k^B) \cup (E_k^B \setminus E_k^A)] = \frac{1}{N} \sum_{u \in \mathcal{U}} |\max(p(u), q(u)) - \min(p(u), q(u))| = \frac{1}{N} \sum_{u \in \mathcal{U}} |p(u) - q(u)|.$$

Since the trials are independent and identically distributed, the probability that $I \neq J$ is the conditional probability that exactly one accepts given that at least one accepts:

$$\Pr[I \neq J] = \frac{\sum_{u \in \mathcal{U}} |p(u) - q(u)|}{\sum_{u \in \mathcal{U}} \max(p(u), q(u))}.$$

Observe that $\sum_u \max(p(u), q(u)) \geq \sum_u p(u) = 1$. Therefore, the denominator is at least 1. We obtain:

$$\Pr[M^A \neq M^B] \leq \Pr[I \neq J] \leq \sum_{u \in \mathcal{U}} |p(u) - q(u)|.$$

The term $\sum |p(u) - q(u)|$ is the L_1 distance between the distributions, which corresponds to $2d_{TV}(p, q)$. Thus, the probability of disagreement is at most $2|p - q|$. □

H.3 Problem 7.3

H.3.1 Problem Statement

Theorem H.3. *Let X, Y be jointly distributed in $\{0,1\}^n$ as follows. Let $I \in [n]$ be uniform, and let (X, Y) be uniform conditioned on $X_{<I} = Y_{<I}$ and $X_I \neq Y_I$.*

1. Compute $\mathsf{I}(X : Y)$.

2. Show that if the parties have shared randomness, Alice gets X as input, Bob gets Y as input, and Alice sends a message M to Bob in a way that allows Bob to deduce the value of X from M, Y then the expected length of M is at least $n/2 - O(1)$.

3. Deduce Corollary 7.7 does not hold without the logarithmic term.

H.3.2 GPT

Proof. We are tasked with proving the given statements from the theorem.

Part 1: Compute $\mathsf{I}(X : Y)$

Let $X, Y \in \{0,1\}^n$ be jointly distributed as described. We know that $I \in [n]$ is uniform, and that (X, Y) is uniform conditioned on $X_{<I} = Y_{<I}$ and $X_I \neq Y_I$. In other words, given that the first $I - 1$ bits of X and Y are identical, and that the I-th bit differs between X and Y, the pair (X, Y) is uniform.

To compute $\mathsf{I}(X : Y)$, we use the definition of mutual information:

$$\mathsf{I}(X : Y) = H(X) - H(X|Y)$$

Since (X, Y) is uniformly distributed conditioned on $X_{<I} = Y_{<I}$ and $X_I \neq Y_I$, we can calculate the entropy of X and Y as follows.

The joint distribution of (X, Y) consists of 2^{n-1} possible configurations due to the uniform distribution over all pairs where $X_{<I} = Y_{<I}$ and $X_I \neq Y_I$. Therefore, the total number of possible outcomes for X and Y is 2^{n-1}, which implies the joint entropy is:

$$H(X, Y) = \log_2(2^{n-1}) = n - 1$$

Next, we compute the conditional entropy $H(X|Y)$. Since $X_I \neq Y_I$ and the first $I - 1$ bits are identical between X and Y, the conditional distribution of X given Y can be deduced from the uniformity over the $n - 1$ bits, except for the I-th bit, which is determined by $X_I \neq Y_I$. Thus, there is no uncertainty about X conditioned on Y, leading to:

$$H(X|Y) = 1$$

Now, we compute the mutual information:

$$\mathsf{I}(X : Y) = H(X) - H(X|Y) = (n - 1) - 1 = n - 2$$

Part 2: Message Length Lower Bound

In this part, we show that if Alice receives X as input and Bob receives Y as input, and Alice sends a message M to Bob such that Bob can deduce X from (M, Y), then the expected length of M is at least $n/2 - O(1)$.

Let X and Y be as described, and let M be the message that Alice sends to Bob. Since Bob can deduce X from (M, Y), it follows that the mutual information between X and the pair (M, Y) is at least $H(X)$, which is the total entropy of X. Thus, we have:

$$\mathsf{I}(X : M, Y) \geq H(X) = n$$

By the data processing inequality, we know that:

$$\mathsf{I}(X : M, Y) \leq H(M)$$

Therefore, we must have:

$$H(M) \geq n$$

Since the expected length of M is the expected entropy of the message, we conclude that the expected length of M must be at least $n/2 - O(1)$.

Part 3: Deduce Corollary 7.7 Does Not Hold Without the Logarithmic Term

Corollary 7.7 asserts that there is a logarithmic term involved in the communication complexity of the problem, typically stemming from the entropy of the system. From the previous calculations, we see that the mutual information $\mathsf{I}(X : Y) = n - 2$ and the expected message length is at least $n/2 - O(1)$, which is a lower bound on the communication complexity. This lower bound does not involve a logarithmic term, and hence Corollary 7.7 does not hold without such a term.

□

H.3.3 Claude

Proof. We prove each part of the theorem in turn.

Part 1: Computing $\mathsf{I}(X : Y)$.

First, we analyze the distribution of (X, Y). Given that $I \in [n]$ is uniform, and conditioned on $I = i$, we have $X_{<i} = Y_{<i}$, $X_i \neq Y_i$, and (X, Y) is uniform subject to these constraints.

For a fixed $i \in [n]$, conditioned on $I = i$:

- The coordinates $X_1, \ldots, X_{i-1}$ equal $Y_1, \ldots, Y_{i-1}$ respectively
- $X_i \neq Y_i$, so $(X_i, Y_i) \in \{(0, 1), (1, 0)\}$ with equal probability
- The remaining coordinates $X_{i+1}, \ldots, X_n$ and $Y_{i+1}, \ldots, Y_n$ are independent and uniform

We compute the mutual information:

$$\mathsf{I}(X : Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$$

Since X is uniform over $\{0, 1\}^n$, we have $H(X) = n$.

To compute $H(X \mid Y)$, we condition on I and Y:

$$H(X \mid Y) = \sum_{i=1}^{n} \Pr[I = i] H(X \mid Y, I = i)$$

Given $I = i$ and Y:

- $X_j = Y_j$ for $j < i$ (determined, contributes 0 to entropy)
- $X_i \neq Y_i$ (1 bit of entropy, since X_i is the unique bit different from Y_i)
- X_j for $j > i$ are independent and uniform (each contributes 1 bit)

Therefore:

$$H(X \mid Y, I = i) = 0 + 1 + (n - i) = n - i + 1$$

Thus:

$$H(X \mid Y) = \frac{1}{n} \sum_{i=1}^{n} (n - i + 1) = \frac{1}{n} \sum_{j=1}^{n} j = \frac{1}{n} \cdot \frac{n(n+1)}{2} = \frac{n+1}{2}$$

Therefore:

$$\mathsf{I}(X : Y) = H(X) - H(X \mid Y) = n - \frac{n+1}{2} = \frac{n-1}{2}$$

Part 2: Lower bound on expected message length.

Suppose Alice receives X, Bob receives Y, they share randomness R, and Alice sends message $M = M(X, R)$ to Bob such that Bob can deduce X from (M, Y, R).

This means $H(X \mid M, Y, R) = 0$. By the data processing inequality and properties of entropy:

$$H(X \mid Y, R) \geq H(X \mid M, Y, R) = 0$$

Since R is independent of (X, Y), we have $H(X \mid Y, R) = H(X \mid Y)$.
The expected message length satisfies:

$$\mathbb{E}[|M|] \geq H(M \mid Y, R)$$

By the chain rule for mutual information:

$$\mathsf{I}(X : M, Y \mid R) = \mathsf{I}(X : Y \mid R) + \mathsf{I}(X : M \mid Y, R)$$

Since R is independent of (X, Y), we have $\mathsf{I}(X : Y \mid R) = \mathsf{I}(X : Y) = \frac{n-1}{2}$.
Since Bob can determine X from (M, Y, R), we have $H(X \mid M, Y, R) = 0$, which implies:

$$\mathsf{I}(X : M, Y \mid R) = H(X \mid R) = H(X) = n$$

Therefore:

$$\mathsf{I}(X : M \mid Y, R) = n - \frac{n-1}{2} = \frac{n+1}{2}$$

Since $\mathsf{I}(X : M \mid Y, R) \leq H(M \mid Y, R) \leq \mathbb{E}[|M|]$, we obtain:

$$\mathbb{E}[|M|] \geq \frac{n+1}{2} = \frac{n}{2} + \frac{1}{2}$$

This gives the lower bound of $\frac{n}{2} - O(1)$ as required.
Part 3: Corollary 7.7 without logarithmic term.
Corollary 7.7 typically states that communication complexity is bounded by mutual information up to logarithmic factors. Specifically, if $\mathsf{I}(X : Y) = I$, then communication complexity is $O(I \log I)$ or similar.
In our example, $\mathsf{I}(X : Y) = \frac{n-1}{2} \approx \frac{n}{2}$, but the communication complexity is at least $\frac{n}{2}$. If Corollary 7.7 held without the logarithmic term (i.e., communication $\leq C \cdot \mathsf{I}(X : Y)$ for constant C), we would have communication at most $C \cdot \frac{n}{2}$.
However, our lower bound shows communication is at least $\frac{n}{2} - O(1)$, which matches the mutual information up to constants. This demonstrates that the logarithmic factor in Corollary 7.7 is necessary: without it, the bound would be tight in this example, but this is a special case. The logarithmic term accounts for the general case where the relationship between mutual information and communication complexity requires additional overhead. □

H.3.4 Gemini

Proof. We address the three parts of the theorem sequentially.
1. Computation of $\mathsf{I}(X : Y)$.
First, we define the probability distribution of (X, Y) explicitly. Let $S_i \subseteq \{0,1\}^n \times \{0,1\}^n$ be the set of pairs (x, y) such that $x_{<i} = y_{<i}$ and $x_i \neq y_i$. The size of this set is determined by the number of choices for the prefix (where x and y agree), the bit at position i (where they differ), and the suffixes (where they are unconstrained).

$$|S_i| = 2^{i-1} \times 2 \times (2^{n-i})^2 = 2^i \cdot 2^{2n-2i} = 2^{2n-i}.$$

The random variable I is uniform on $[n]$, so $\Pr[I = i] = \frac{1}{n}$. Conditioned on $I = i$, (X, Y) is uniform on S_i. Since the sets S_i are disjoint for distinct i (as the index of the first bit difference is unique for any $x \neq y$), the joint probability mass function is:

$$p(x, y) = \sum_{i=1}^{n} \Pr[I = i] \cdot p(x, y | I = i) = \begin{cases} \frac{1}{n} \cdot \frac{1}{2^{2n-i}} & \text{if } (x, y) \in S_i \\ 0 & \text{if } x = y \end{cases}.$$

To compute the mutual information $\mathsf{I}(X:Y) = H(X) + H(Y) - H(X,Y)$, we first find the marginal entropies. Consider the marginal distribution of X. For any fixed $x \in \{0,1\}^n$:

$$p(x) = \sum_y p(x,y) = \sum_{i=1}^{n} \sum_{y:(x,y)\in S_i} \frac{1}{n2^{2n-i}}.$$

For a fixed x and i, the number of y such that $(x,y) \in S_i$ is the number of y such that $y_{<i} = x_{<i}$, $y_i \neq x_i$, and $y_{>i}$ is arbitrary. There are $1 \cdot 1 \cdot 2^{n-i}$ such y's. Thus:

$$p(x) = \sum_{i=1}^{n} \frac{2^{n-i}}{n2^{2n-i}} = \sum_{i=1}^{n} \frac{1}{n2^n} = \frac{1}{2^n}.$$

So X is uniformly distributed on $\{0,1\}^n$, and $H(X) = n$. By symmetry, Y is also uniform, so $H(Y) = n$.

Next, we compute the joint entropy $H(X,Y)$. Since the pair (X,Y) uniquely determines the index of the first difference I (whenever $X \neq Y$, which occurs with probability 1), we have $H(X,Y) = H(X,Y,I)$. Using the chain rule:

$$H(X,Y) = H(I) + H(X,Y|I).$$

Since I is uniform on $[n]$, $H(I) = \log n$. For the conditional entropy:

$$H(X,Y|I) = \sum_{i=1}^{n} \Pr[I=i] H(X,Y|I=i).$$

Given $I = i$, (X,Y) is uniform on S_i, which has size 2^{2n-i}. Thus, $H(X,Y|I=i) = \log(2^{2n-i}) = 2n - i$.

$$H(X,Y|I) = \frac{1}{n}\sum_{i=1}^{n}(2n-i) = 2n - \frac{1}{n}\sum_{i=1}^{n} i = 2n - \frac{n(n+1)}{2n} = 2n - \frac{n+1}{2}.$$

Therefore, $H(X,Y) = \log n + 2n - \frac{n+1}{2}$.

Finally, the mutual information is:

$$\mathsf{I}(X:Y) = H(X) + H(Y) - H(X,Y) = n + n - \left(\log n + 2n - \frac{n+1}{2}\right) = \frac{n+1}{2} - \log n.$$

2. Lower bound on expected message length.

Let M be the message sent by Alice. Bob possesses Y and shares randomness R with Alice. Bob must compute X from (M,Y,R). This is a source coding problem with side information. A fundamental lower bound on the expected length $\mathbb{E}[|M|]$ of any prefix-free code (or uniquely decodable code) allowing perfect reconstruction of X given Y is the conditional entropy $H(X|Y)$. Even with shared randomness independent of (X,Y), the bound holds:

$$\mathbb{E}[|M|] \geq H(X|Y).$$

We compute $H(X|Y)$:

$$H(X|Y) = H(X,Y) - H(Y) = \left(\log n + 2n - \frac{n+1}{2}\right) - n = n + \log n - \frac{n}{2} - \frac{1}{2} = \frac{n}{2} + \log n - \frac{1}{2}.$$

Thus,

$$\mathbb{E}[|M|] \geq \frac{n}{2} + \log n - \frac{1}{2}.$$

Since $\log n \geq 0$ for $n \geq 1$, we have $\mathbb{E}[|M|] \geq \frac{n}{2} - O(1)$. Specifically, the length is at least $\frac{n}{2} + \Omega(\log n)$.

3. Deduction regarding Corollary 7.7.

We compare the mutual information $\mathsf{I}(X : Y)$ with the communication cost (expected length of M). From Part 1:

$$\mathsf{I}(X : Y) = \frac{n}{2} - \log n + \frac{1}{2}.$$

From Part 2, the necessary communication L satisfies:

$$L \geq \frac{n}{2} + \log n - \frac{1}{2}.$$

The difference between the communication cost and the mutual information is:

$$L - \mathsf{I}(X : Y) \geq \left(\frac{n}{2} + \log n - \frac{1}{2}\right) - \left(\frac{n}{2} - \log n + \frac{1}{2}\right) = 2\log n - 1.$$

If Corollary 7.7 posits a bound of the form $L \leq \mathsf{I}(X : Y) + O(1)$ (i.e., communication equals information up to a constant additive term), this example serves as a counterexample, as the gap grows logarithmically with n. Therefore, Corollary 7.7 cannot hold without including a logarithmic term in the bound relating communication complexity to information complexity. □

H.4 Problem 7.4

H.4.1 Problem Statement

Theorem H.4. *Alice and Bob can use public randomness to simulate sending M with expected communication $\mathsf{I}(X : M) + 2\log(1 + \mathsf{I}(X : M)) + O(1)$. The simulation is one round, external, and without error.*

H.4.2 GPT

Proof. We are tasked with proving that Alice and Bob can use public randomness to simulate sending a message M with expected communication $\mathsf{I}(X : M) + 2\log(1 + \mathsf{I}(X : M)) + O(1)$, where the simulation is one round, external, and without error. To achieve this, we will analyze the communication required in terms of mutual information and the role of public randomness in this simulation.

Let X be the random variable representing the public randomness available to Alice and Bob. The mutual information between X and M, denoted as $\mathsf{I}(X : M)$, measures the amount of information that M conveys given the randomness X. We assume that Alice and Bob can agree on a protocol to utilize this public randomness in a way that efficiently simulates the communication of M.

First, note that the mutual information $\mathsf{I}(X : M)$ captures the fundamental communication cost to simulate the transmission of M under the given conditions. However, in the case of public randomness, the actual communication required may differ due to the potential for shared knowledge between Alice and Bob.

To refine the communication bound, we use the fact that public randomness allows Alice and Bob to correlate their choices without explicitly transmitting all of the information. This results in a reduction in the required communication. Specifically, the additional communication cost is related to the entropy of the public randomness, and we expect an extra term of $2\log(1 + \mathsf{I}(X : M))$ to account for this.

The $O(1)$ term represents a constant overhead that arises due to the finite precision required in the communication and the overhead of the protocol. This term does not grow with the size of the message M and is therefore considered negligible in the asymptotic analysis.

Thus, the total expected communication for the simulation can be expressed as:

$$\mathsf{I}(X : M) + 2\log(1 + \mathsf{I}(X : M)) + O(1)$$

To verify the correctness of this result, observe that the mutual information term $\mathsf{I}(X : M)$ corresponds to the minimal communication required if there were no public randomness. The additional term $2\log(1 + \mathsf{I}(X : M))$ accounts for the potential gains in communication efficiency due to the use of public randomness, and the $O(1)$ term reflects a constant overhead in the communication process.

Since the simulation is one round, external, and without error, the total communication cost remains within the expected bound. Therefore, we conclude that Alice and Bob can indeed simulate the sending of M with the expected communication cost of:

$$\mathsf{I}(X : M) + 2\log(1 + \mathsf{I}(X : M)) + O(1)$$

as desired.

□

H.4.3 Claude

Proof. We construct a protocol that allows Alice and Bob to simulate the transmission of message M using public randomness with the stated expected communication cost.

Let X denote Alice's input and M denote the message to be transmitted. We assume that Alice and Bob have access to shared public randomness.

The protocol proceeds as follows:

Alice and Bob use a randomized encoding scheme based on the distribution of M conditioned on X. Define $I = \mathsf{I}(X : M)$ to be the mutual information between X and M.

By the source coding theorem with side information, there exists a code that allows Alice to communicate M to Bob (who knows X) using expected length approximately $\mathsf{H}(M \mid X)$ bits. Since $\mathsf{I}(X : M) = \mathsf{H}(M) - \mathsf{H}(M \mid X)$, we have $\mathsf{H}(M \mid X) = \mathsf{H}(M) - I$.

However, we employ a more sophisticated approach using rejection sampling and public randomness. Alice and Bob share access to a public random string R that is independent of both X and M.

Alice uses the public randomness R to generate a random sample from the conditional distribution $P_{M|X}(\cdot \mid X)$. She compares this sample with the actual message M. If they match, she sends a short confirmation message. If they do not match, she continues sampling.

More precisely, Alice partitions the message space into bins using a hash function derived from the public randomness. The number of bins is chosen to be approximately 2^I, where $I = \mathsf{I}(X : M)$.

Alice sends the bin index B containing M, which requires $\log(2^I) = I$ bits. Additionally, she sends the position of M within the bin, which requires on average $\log(1 + I)$ bits due to the structure of the conditional distribution.

To handle edge cases and ensure exact simulation without error, Alice includes a refinement message of length $\log(1 + I) + O(1)$ bits that uniquely identifies M within its bin.

The total expected communication is:

$$\mathsf{I}(X : M) + 2\log(1 + \mathsf{I}(X : M)) + O(1).$$

The protocol is one-round because Alice sends a single message to Bob. It is external because the public randomness is independent of the inputs. It is without error because Bob can perfectly reconstruct M from Alice's message and the public randomness.

By the properties of conditional entropy and the careful construction of the encoding scheme using public randomness, the expected communication complexity achieves the stated bound. □

H.4.4 Gemini

Proof. Let $P_{M|X}$ denote the conditional distribution of M given X, and let $Q_M(m) = \sum_x P_X(x)P_{M|X}(m|x)$ be the marginal distribution of M. We construct a protocol using shared randomness based on the Poisson Point Process (PPP) method for rejection sampling, often referred to as the Braverman-Rao protocol.

Shared Randomness: Alice and Bob share an infinite sequence of points $\mathcal{P} = \{(M_i, T_i)\}_{i\geq 1} \subset \mathcal{M} \times \mathbb{R}_{\geq 0}$, generated according to a Poisson Point Process with intensity measure $\mu(dm, dt) = Q_M(m)\, dt$. Constructively, this can be realized by letting $M_i \sim Q_M$ be i.i.d. samples and T_i be the arrival times of a standard Poisson process (i.e., $T_i = \sum_{k=1}^{i} E_k$ where $E_k \sim \text{Exp}(1)$). The pairs are indexed such that $0 < T_1 < T_2 < \dots$.

Protocol:

1. Alice and Bob receive inputs. Alice has x.

2. Alice computes the values $V_i = \frac{T_i}{w(M_i|x)}$ for all $i \geq 1$, where $w(m|x) = \frac{P_{M|X}(m|x)}{Q_M(m)}$ is the density ratio.

3. Alice identifies the index J that minimizes this value:

$$J = \arg\min_{i \geq 1} V_i = \arg\min_{i \geq 1} \frac{T_i \cdot Q_M(M_i)}{P_{M|X}(M_i|x)}.$$

4. Alice encodes the integer J using Elias δ-coding and sends the codeword to Bob.

5. Bob decodes J and outputs M_J.

Correctness: Consider the transformation of the PPP $\mathcal{P}$ under the mapping $(m, t) \mapsto (m, v)$ where $v = t/w(m|x)$. The intensity measure of the transformed process $\mathcal{P}'$ is given by:

$$\lambda'(m, v)\, dm\, dv = Q_M(m)\, d(v \cdot w(m|x)) = Q_M(m) \frac{P_{M|X}(m|x)}{Q_M(m)}\, dv = P_{M|X}(m|x)\, dv.$$

Let $\mathcal{V} = \{V_i\}_{i \geq 1}$ be the projection of $\mathcal{P}'$ onto the second coordinate. The points in $\mathcal{V}$ form a Poisson process on $\mathbb{R}_{\geq 0}$ with intensity $\sum_m P_{M|X}(m|x) = 1$. Thus, the minimum value $V_{\min} = \min_i V_i$ follows an exponential distribution $\mathrm{Exp}(1)$. The element M_J corresponding to the minimum $V_{\min}$ is selected with probability proportional to the intensity at that point. Specifically, conditioned on $V_{\min} = v$, the probability that the minimizing mark is m is proportional to $P_{M|X}(m|x)$. Therefore, the marginal distribution of the output M_J is exactly $P_{M|X}(\cdot|x)$. The simulation is zero-error.

Communication Cost: The communication cost is the length of the Elias δ-code for J, denoted $L(J)$. We have $L(J) = \log J + 2\log(\log J + 1) + O(1)$. We analyze the expected value $\mathbb{E}[L(J)]$. Conditioned on the choice of $M_J = m$ and the minimum value $V_{\min} = v$, the index J represents the position of the point (m, T_J) in the ordered sequence of arrival times $T_1 < T_2 < \ldots$. Specifically, J is the number of points (M_k, T_k) in the original process such that $T_k \leq T_J$. We have $T_J = V_{\min} \cdot w(M_J|x)$. Since the original process has unit intensity over $\mathcal{M}$, the count of points in the interval $[0, T_J]$ (excluding the point J itself) follows a Poisson distribution with mean T_J. Thus, $\mathbb{E}[J \mid T_J] = T_J + 1 \approx T_J$. Using the approximation $\log J \approx \log T_J$ (which can be formalized using Jensen's inequality and properties of the Poisson distribution to show $\mathbb{E}[\log J] \leq \mathbb{E}[\log T_J] + O(1)$), we consider $\mathbb{E}[\log T_J]$.

$$\log T_J = \log V_{\min} + \log w(M_J|x) = \log V_{\min} + \log \frac{P_{M|X}(M_J|x)}{Q_M(M_J)}.$$

Taking expectations over the randomness of the protocol for a fixed x:

$$\begin{aligned}\mathbb{E}[\log T_J \mid X = x] &= \mathbb{E}[\log V_{\min}] + \mathbb{E}_{M \sim P_{M|X=x}}\left[\log \frac{P_{M|X}(M|x)}{Q_M(M)}\right] \\ &= -\gamma + D(P_{M|X=x} \| Q_M),\end{aligned}$$

where γ is the Euler-Mascheroni constant and $D(\cdot\|\cdot)$ is the Kullback-Leibler divergence. Thus, $\mathbb{E}[\log J \mid X = x] \leq D(P_{M|X=x}\|Q_M) + O(1)$.

Now, consider the second-order term in the coding length:

$$\mathbb{E}[\log(\log J + 1) \mid X = x] \leq \log(\mathbb{E}[\log J \mid X = x] + 1) \leq \log(D(P_{M|X=x}\|Q_M) + O(1)).$$

The total expected communication for input x is:

$$C(x) \leq D(P_{M|X=x}\|Q_M) + 2\log(1 + D(P_{M|X=x}\|Q_M)) + O(1).$$

Taking the expectation over X:

$$\mathbb{E}_X[C(X)] \leq \mathbb{E}_X[D(P_{M|X} \| Q_M)] + 2\mathbb{E}_X[\log(1 + D(P_{M|X} \| Q_M))] + O(1).$$

By definition, $\mathbb{E}_X[D(P_{M|X} \| Q_M)] = \mathsf{I}(X : M)$. Using Jensen's inequality for the concave function $f(z) = \log(1 + z)$:

$$\mathbb{E}_X[\log(1 + D(P_{M|X} \| Q_M))] \leq \log(1 + \mathbb{E}_X[D(P_{M|X} \| Q_M)]) = \log(1 + \mathsf{I}(X : M)).$$

Combining these terms, the total expected communication is bounded by:

$$\mathsf{I}(X : M) + 2\log(1 + \mathsf{I}(X : M)) + O(1).$$

This completes the proof. □

H.5 Problem 7.5

H.5.1 Problem Statement

Lemma H.5. *Alice can send Bob any integer using a protocol that communicates* $\log z + 2\log\log z + O(1)$ *bits to send the integer* z.

H.5.2 GPT

Proof. We are given that Alice can send Bob any integer z using a protocol that communicates $\log z + 2\log\log z + O(1)$ bits. Our goal is to prove the correctness of this claim.

First, let us consider the task of encoding an integer z in binary. The number of bits required to represent z is $\lceil \log_2 z \rceil$ bits. Therefore, the most straightforward encoding of z requires $\log_2 z$ bits, which is $O(\log z)$ bits.

Next, we observe that a more efficient encoding can be achieved by applying a method such as prefix coding, which reduces the total number of bits required for certain types of integers. A well-known approach for encoding integers efficiently is to use a combination of unary and binary coding schemes. Specifically, for large integers z, the number of bits required can be reduced by encoding the logarithm of z.

To make this precise, consider the encoding that first transmits $\log z$ bits (to communicate the magnitude of z), followed by an additional $2\log\log z$ bits that encode the number of bits needed to represent the logarithmic component of z. This results in a total communication cost of:

$$\log z + 2\log\log z$$

Finally, there is an $O(1)$ term that accounts for any constant overhead in the protocol, such as the prefix or structure of the encoding.

Therefore, the total number of bits communicated is given by:

$$\log z + 2\log\log z + O(1)$$

This proves that Alice can send Bob any integer using a protocol that communicates $\log z + 2\log\log z + O(1)$ bits. Thus, the lemma is proven. □

H.5.3 Claude

Proof. We construct an explicit protocol for Alice to send an arbitrary integer z to Bob using $\log z + 2\log\log z + O(1)$ bits.

Without loss of generality, assume $z > 0$ is a positive integer (the sign can be encoded using 1 additional bit). The protocol proceeds in two phases.

Phase 1: Communicating the bit-length.

Let $n = \lfloor \log_2 z \rfloor + 1$ denote the number of bits required to represent z in binary. Note that $n \leq \log_2 z + 1$, so $\log_2 n \leq \log_2(\log_2 z + 1) = \log_2 \log_2 z + O(1)$.

Alice first communicates n to Bob using the following self-delimiting encoding:

- Let $m = \lfloor \log_2 n \rfloor + 1$ be the bit-length of n.
- Alice sends m in unary: she transmits $m - 1$ zeros followed by a single one, using m bits total.
- Alice then sends the binary representation of n (excluding the leading 1, which is implicit), using $m - 1$ bits.

The total number of bits used in Phase 1 is:

$$m + (m - 1) = 2m - 1 = 2\lfloor \log_2 n \rfloor + 1 \leq 2 \log_2 n + 1 \leq 2 \log_2 \log_2 z + O(1)$$

Phase 2: Communicating the value.
Once Bob knows n, Alice sends the n-bit binary representation of z, using exactly n bits. Since $n = \lfloor \log_2 z \rfloor + 1$, we have:

$$n \leq \log_2 z + 1$$

Total communication cost.
The total number of bits transmitted is:

$$\text{Phase 1} + \text{Phase 2} = (2 \log_2 n + O(1)) + n$$

Substituting $n \leq \log_2 z + 1$ and $\log_2 n \leq \log_2 \log_2 z + O(1)$:

$$\text{Total bits} \leq (\log_2 z + 1) + 2(\log_2 \log_2 z + O(1)) + O(1) = \log_2 z + 2 \log_2 \log_2 z + O(1)$$

Converting to natural logarithm (noting that $\log_2 x = \frac{\log x}{\log 2}$ differs only by a constant factor):

$$\text{Total bits} = \log z + 2 \log \log z + O(1)$$

This completes the construction of the protocol, establishing the lemma. □

H.5.4 Gemini

Proof. We construct a self-delimiting prefix code for positive integers, specifically the Elias δ-code, and analyze its length. For an arbitrary integer z, we may assume without loss of generality that z is positive, as a sign bit adds only 1 bit of overhead, which is absorbed into the $O(1)$ term. Alternatively, we can map integers $\mathbb{Z}$ to positive integers $\mathbb{Z}^+$ via a bijection such as $z \mapsto 2|z|$ if $z > 0$ and $z \mapsto 2|z| + 1$ if $z \leq 0$, which increases the logarithm of the value by at most 1 bit.

Let $b(n)$ denote the standard binary representation of a positive integer n, and let $|b(n)| = \lfloor \log_2 n \rfloor + 1$ denote its length. We define the encoding of an integer $z \geq 1$ in three parts.

First, let $N = \lfloor \log_2 z \rfloor$. This N represents the number of bits required to encode z excluding the leading most significant bit (which is always 1). To allow the receiver to parse the stream correctly, we must encode the length N. Since N can be large, we encode N (or rather $N + 1$ to handle $N = 0$) using a variable-length code.

Let $L = \lfloor \log_2(N + 1) \rfloor$. We encode the value $N + 1$ using the Elias γ-code. This consists of a unary representation of L followed by the binary representation of $N + 1$. Specifically, we transmit L zeros, followed by a single 1, followed by the L least significant bits of $N + 1$. The length of this preamble is $L + 1 + L = 2L + 1$ bits.

Finally, we append the N least significant bits of z. The total length of the code $C(z)$ is given by the sum of the lengths of these components:

$$|C(z)| = (2\lfloor \log_2(N + 1) \rfloor + 1) + N.$$

Substituting $N = \lfloor \log_2 z \rfloor$, we obtain:

$$|C(z)| = 2\lfloor \log_2(\lfloor \log_2 z \rfloor + 1) \rfloor + 1 + \lfloor \log_2 z \rfloor.$$

We apply the inequality $x - 1 < \lfloor x \rfloor \leq x$. Thus, $\lfloor \log_2 z \rfloor \leq \log_2 z$ and $\lfloor \log_2(\lfloor \log_2 z \rfloor + 1) \rfloor \leq \log_2(\log_2 z + 1)$. For $z \geq 2$, $\log_2 z + 1 \leq 2 \log_2 z$, implying $\log_2(\log_2 z + 1) \leq \log_2(2 \log_2 z) = \log_2 \log_2 z + 1$.

Substituting these bounds into the length expression yields:

$$|C(z)| \leq 2(\log_2 \log_2 z + 1) + 1 + \log_2 z = \log_2 z + 2 \log_2 \log_2 z + 3.$$

Since 3 is a constant, the total number of bits communicated is $\log z + 2 \log \log z + O(1)$. The decoding process is unique and instantaneous: the receiver counts the initial zeros to determine L, reads the next $L+1$ bits to reconstruct $N+1$ (and thus N), and finally reads the subsequent N bits to reconstruct z. □

H.6 Problem 7.6

H.6.1 Problem Statement

Theorem H.6. *Suppose Alice knows a distribution p over the set $\mathcal{U}$, and Bob knows q. For every $\epsilon > 0$, there is a protocol for Alice to sample an element according to the distribution p while communicating at most*

$$\mathsf{D}(p\|q) + O\left(\sqrt{\mathsf{D}(p\|q)}\right) + \log(1/\epsilon) + O(1)$$

H.6.2 GPT

Proof. Let p and q be two probability distributions over the same set $\mathcal{U}$. Alice knows p and Bob knows q. The goal is to show that for every $\epsilon > 0$, there exists a protocol by which Alice can sample from p while communicating at most

$$\mathsf{D}(p\|q) + O\left(\sqrt{\mathsf{D}(p\|q)}\right) + \log\left(\frac{1}{\epsilon}\right) + O(1),$$

where $\mathsf{D}(p\|q)$ is the Kullback-Leibler divergence between p and q.

Step 1: Setup and assumptions We assume that Alice has access to the distribution p and Bob has access to q, where $\mathsf{D}(p\|q)$ is finite. The task is for Alice to sample from p while communicating as efficiently as possible with Bob. Denote the message that Alice sends to Bob as m, and let the probability of Alice sending a particular message be governed by p_m.

Step 2: Communication cost and strategy Alice can send a message m to Bob based on her distribution p. Bob, after receiving the message, can apply a decoding procedure to generate a sample from p. The communication cost depends on the mutual information between the message and the desired sample. The mutual information I between Alice's message and Bob's sample is bounded by the Kullback-Leibler divergence $\mathsf{D}(p\|q)$.

It is known from information theory that the mutual information can be bounded as follows:

$$I(m; X) \leq \mathsf{D}(p\|q).$$

This provides the first term in the communication cost, which is $O(\mathsf{D}(p\|q))$.

Step 3: Communication efficiency using coding To improve communication efficiency, we use a coding strategy based on the distribution q. The basic idea is that Alice can sample from p using a coding scheme that allows Bob to decode the sample with a small communication overhead. By using a universal coding scheme, Alice can send a message that encodes the sample from p with a communication cost related to the Kullback-Leibler divergence.

Through standard coding arguments (such as universal coding and entropy bounds), Alice can communicate at a rate close to $\mathsf{D}(p\|q)$. However, due to the randomness inherent in sampling, a slight overhead is necessary to ensure that the sample is distributed according to p. This overhead is proportional to $\sqrt{\mathsf{D}(p\|q)}$, resulting in an additional term $O(\sqrt{\mathsf{D}(p\|q)})$.

Step 4: Handling error probability To ensure that Alice can generate the sample from p with high probability, we must account for the error probability. The total error probability can be controlled by adding a term related to the logarithm of $1/\epsilon$. In particular, using standard results from information theory, the error probability can be reduced by sending an additional amount of communication proportional to $\log(1/\epsilon)$. This ensures that the error probability is bounded by ϵ.

Step 5: Final communication cost Combining these terms, the total communication cost for Alice to sample from p is given by:

$$\mathsf{D}(p\|q) + O\left(\sqrt{\mathsf{D}(p\|q)}\right) + \log\left(\frac{1}{\epsilon}\right) + O(1),$$

which completes the proof.

Thus, we have shown that there exists a protocol for Alice to sample from p while communicating at most the specified amount of information. □

H.6.3 Claude

Proof. We construct a protocol based on rejection sampling with communication-efficient encoding.

Let $\epsilon > 0$ be given. Define $M = \lceil \exp(\mathsf{D}(p\|q)) \rceil$ and set $\delta = \epsilon/M$.

The protocol proceeds as follows:

Alice and Bob first agree on a threshold parameter $T = \mathsf{D}(p\|q) + c\sqrt{\mathsf{D}(p\|q)} + \log(1/\delta)$ for an appropriate constant $c > 0$ to be determined.

In each round $i = 1, 2, \ldots$:

Bob samples $X_i \sim q$ and computes the likelihood ratio $w_i = p(X_i)/q(X_i)$. Bob sends to Alice a binary indicator b_i where $b_i = 1$ if $-\log w_i \leq T$ and $b_i = 0$ otherwise.

If $b_i = 0$, the round is rejected and they proceed to round $i + 1$.

If $b_i = 1$, Bob encodes X_i using an optimal prefix-free code with expected length at most $H_q(X \mid -\log w \leq T) + 1$, where H_q denotes entropy under distribution q.

Alice receives the encoding, decodes to obtain X_i, and accepts X_i with probability $w_i/\exp(T)$. If accepted, output X_i and terminate. Otherwise, proceed to round $i + 1$.

We now analyze the correctness and communication cost.

For correctness, observe that the probability Alice outputs element x is proportional to $p(x) \cdot \mathbb{1}[-\log(p(x)/q(x)) \leq T]$. By choosing T sufficiently large, we ensure $\mathbb{P}_p[-\log(p/q) > T] < \epsilon$, so the output distribution is ϵ-close to p in total variation distance.

Specifically, by Markov's inequality applied to the random variable $-\log(p/q)$ under distribution p:

$$\mathbb{P}_p\left[-\log(p/q) - \mathsf{D}(p\|q) > t\right] \leq \frac{\mathrm{Var}_p(-\log(p/q))}{t^2}$$

Since $\mathrm{Var}_p(-\log(p/q)) \leq \mathbb{E}_p[(\log(p/q))^2] = O(\mathsf{D}(p\|q))$ for distributions with bounded support or appropriate tail conditions, setting $t = c\sqrt{\mathsf{D}(p\|q)}$ with c large enough ensures the tail probability is at most $\epsilon/2$.

For the communication cost, the expected number of rounds until acceptance is at most $\exp(T)/\mathbb{E}_q[\min(w, \exp(T))]$. By the definition of KL divergence and Jensen's inequality, this is bounded by $O(1)$ when $T = \mathsf{D}(p\|q) + O(\sqrt{\mathsf{D}(p\|q)}) + \log(1/\epsilon)$.

The communication per round consists of: - One bit for the indicator b_i - When $b_i = 1$, the encoding of X_i with expected length bounded by $H_q(X \mid -\log w \leq T) + 1$

By the properties of conditional entropy and the choice of T:

$$H_q(X \mid -\log w \leq T) \leq H_q(X) \leq \mathsf{D}(p\|q) + H_p(X)$$

The total expected communication is therefore:

$$O(1) \cdot (1 + \mathsf{D}(p\|q) + O(\sqrt{\mathsf{D}(p\|q)}) + \log(1/\epsilon) + O(1))$$

Simplifying, the communication complexity is at most:

$$\mathsf{D}(p\|q) + O(\sqrt{\mathsf{D}(p\|q)}) + \log(1/\epsilon) + O(1)$$

as required. □

H.6.4 Gemini

Proof. Let $\mathcal{U}$ be the domain of the distributions p and q. We define the information density random variable $L(x) = \log \frac{p(x)}{q(x)}$. The Kullback-Leibler divergence is given by $\mathsf{D}(p\|q) = \mathbb{E}_{x\sim p}[L(x)]$.

We construct a protocol based on rejection sampling with a cutoff. Let $\tau > 0$ be a threshold parameter to be determined later. Define the set of "typical" elements $S \subseteq \mathcal{U}$ as:

$$S = \{x \in \mathcal{U} : p(x) \leq 2^\tau q(x)\}.$$

We define a sub-normalized distribution p' supported on S by $p'(x) = p(x)$ for $x \in S$ and $p'(x) = 0$ otherwise.

Protocol: Alice and Bob have access to a sequence of shared public random variables $(X_1, Y_1), (X_2, Y_2), \ldots$, where each X_i is drawn independently from q and each Y_i is drawn independently from the uniform distribution on $[0, 1]$. Alice scans the sequence to find the first index $J \geq 1$ such that:

$$Y_J \leq \frac{p'(X_J)}{2^\tau q(X_J)}.$$

Note that since $x \in S \implies p'(x) \leq 2^\tau q(x)$, the term on the right-hand side is in $[0, 1]$. If such an index J is found, Alice sends J to Bob using a prefix-free code. Bob outputs X_J. If no such index is found within a cutoff limit K (defined below), the protocol aborts.

Analysis of Correctness: Conditioned on the event that index J is selected, the probability that $X_J = x$ is:

$$\Pr(X_J = x \mid J \text{ is selected}) = \frac{q(x) \cdot \frac{p'(x)}{2^\tau q(x)}}{\sum_{z\in\mathcal{U}} q(z) \frac{p'(z)}{2^\tau q(z)}} = \frac{p'(x)}{\sum_{z\in S} p(z)} = \frac{p'(x)}{p(S)}.$$

The total variation distance between the output distribution and p is bounded by the probability mass of the set S^c (the "bad" set) plus the probability of aborting.

$$\|p - p_{\text{out}}\|_{\text{TV}} \leq p(S^c) + \Pr(\text{Abort}).$$

We require the total error to be at most ϵ. We allocate $\epsilon/2$ to $p(S^c)$ and $\epsilon/2$ to the abort probability.

Bounding the Tail Probability: We need to choose τ such that $p(S^c) = \Pr_{x\sim p}\left(\log \frac{p(x)}{q(x)} > \tau\right) \leq \frac{\epsilon}{2}$. By the properties of the information density, specifically applying Chebyshev's inequality (or noting the concentration of measure for the information density around the relative entropy), there exists a constant C such that setting

$$\tau = \mathsf{D}(p\|q) + C\sqrt{\mathsf{D}(p\|q)} + \log(2/\epsilon)$$

ensures that $p(S^c) \leq \epsilon/2$. The term $O(\sqrt{\mathsf{D}(p\|q)})$ accounts for the dispersion of the information density.

Communication Cost: The probability of acceptance in a single trial is:

$$\alpha = \mathbb{E}_{x\sim q}\left[\frac{p'(x)}{2^\tau q(x)}\right] = \frac{1}{2^\tau}\sum_{x\in S} p(x) = \frac{p(S)}{2^\tau}.$$

Since $p(S) \geq 1 - \epsilon/2$, we have $\alpha \approx 2^{-\tau}$. The index J follows a geometric distribution with parameter α. The expected value is $\mathbb{E}[J] = 1/\alpha = 2^\tau/p(S)$. Alice sends J to Bob. Using a standard prefix code for integers, the communication cost is approximately $\log J$. To ensure the worst-case communication bound (or bound with high probability), we set a cutoff $K = \frac{2}{\epsilon\alpha}$. By Markov's inequality, the probability that $J > K$ is at most $\epsilon/2$. Thus, with probability at least $1 - \epsilon$, the protocol succeeds. The communication cost is bounded by:

$$\log K = \log\left(\frac{2 \cdot 2^\tau}{\epsilon \cdot p(S)}\right) \approx \tau + \log(1/\epsilon) + O(1).$$

Substituting τ:

$$\text{Cost} \approx \mathsf{D}(p\|q) + O\left(\sqrt{\mathsf{D}(p\|q)}\right) + \log(1/\epsilon) + \log(1/\epsilon) + O(1).$$

Simplifying the terms, the communication is bounded by:

$$\mathsf{D}(p\|q) + O\left(\sqrt{\mathsf{D}(p\|q)}\right) + \log(1/\epsilon) + O(1).$$

This completes the proof. □

H.7 Problem 7.7

H.7.1 Problem Statement

Theorem H.7. *Let X, Y be jointly distributed random variables. There exists a random variable Z such that: (1) Z is independent of (X, Y); (2) Conditioned on $Z = z$, the variable Y becomes a deterministic function of X; and (3) $H(Y \mid Z) \leq I(X : Y) + 2\log I(X : Y) + O(1)$.*

H.7.2 GPT

Proof. Let X and Y be jointly distributed random variables, and we aim to show the existence of a random variable Z satisfying the conditions stated in the theorem.

Existence of the Random Variable Z We begin by defining the random variable Z through a construction based on the mutual information between X and Y. Let Z be a random variable defined as follows:

$$Z = f(X, Y)$$

where f is a function that is designed to satisfy the required properties. Specifically, we will choose f such that Z is independent of (X, Y) and satisfies the other conditions in the theorem.

Property (1): Independence of Z and (X, Y) The construction of Z ensures that Z is independent of both X and Y. By definition, if Z is independent of (X, Y), then the joint distribution of Z and (X, Y) factorizes as follows:

$$p_{Z,X,Y}(z, x, y) = p_Z(z)p_{X,Y}(x, y)$$

This implies that knowing the value of Z does not provide any additional information about the values of X and Y. Hence, Z is independent of (X, Y), which satisfies the first condition of the theorem.

Property (2): Deterministic Function of X Given $Z = z$ Next, we examine the second condition, which states that conditioned on $Z = z$, the variable Y becomes a deterministic function of X. To show this, we condition on the value of $Z = z$ and write the conditional distribution of Y given $Z = z$:

$$p_{Y|Z}(y \mid z) = p_{Y|X,Z}(y \mid x, z)$$

Since Z is constructed in such a way that it encapsulates the information required to determine Y from X, the conditional distribution of Y given $Z = z$ is degenerate. That is, for each z, there exists a deterministic function g_z such that:

$$Y = g_z(X)$$

Therefore, conditioned on $Z = z$, Y is completely determined by X, satisfying the second condition of the theorem.

Property (3): Entropy Bound We now prove the third condition, which involves the entropy of Y conditioned on Z and the mutual information $I(X : Y)$. By the chain rule for mutual information, we have:

$$I(X : Y) = H(Y) - H(Y \mid X)$$

We want to show that:

$$H(Y \mid Z) \leq I(X:Y) + 2\log I(X:Y) + O(1)$$

Using the fact that Y becomes a deterministic function of X given Z, the conditional entropy $H(Y \mid Z)$ is reduced to the entropy of the noise introduced by the conditional independence structure. Thus, we expect that:

$$H(Y \mid Z) \leq I(X:Y) + O(1)$$

The additional term $2\log I(X:Y)$ arises from considerations of the complexity of the function f used in the construction of Z, and from the fact that the mutual information provides a bound on the uncertainty in Y conditioned on X. Finally, the term $O(1)$ represents a small constant that accounts for higher-order terms not captured by the mutual information.

Thus, we have shown that:

$$H(Y \mid Z) \leq I(X:Y) + 2\log I(X:Y) + O(1)$$

which completes the proof of the third condition.

Conclusion We have constructed the random variable Z that satisfies all three conditions: 1. Z is independent of (X, Y), 2. Conditioned on $Z = z$, Y is a deterministic function of X, 3. $H(Y \mid Z) \leq I(X:Y) + 2\log I(X:Y) + O(1)$.

This completes the proof. □

H.7.3 Claude

Proof. Let X and Y be jointly distributed random variables with joint distribution P_{XY}. We construct the random variable Z and establish the three required properties.

Construction of Z: Define Z to be a random variable that takes values in a finite set $\mathcal{Z}$ according to a distribution P_Z that is independent of (X, Y). Specifically, we construct Z using a randomized quantization scheme based on the mutual information $I(X:Y)$.

Let $I = I(X:Y)$ denote the mutual information between X and Y. We may assume $I > 0$, as otherwise X and Y are independent and the result is trivial. Define $M = \lceil 2^{I+2\log I+c} \rceil$ for some sufficiently large constant $c > 0$.

For each $x \in \mathcal{X}$, consider the conditional distribution $P_{Y|X=x}$. We construct a random quantization by selecting M representative points uniformly at random from an appropriate covering of the space of conditional distributions. Let Z be uniformly distributed over $\{1, 2, \ldots, M\}$, independent of (X, Y).

For each value $z \in \{1, 2, \ldots, M\}$, we define a deterministic function $f_z : \mathcal{X} \to \mathcal{Y}$ as follows. Using rate-distortion theory, we know that for any $\epsilon > 0$, there exists a quantization scheme such that we can represent the conditional distribution $P_{Y|X}$ using approximately $I(X:Y)$ bits on average.

Verification of Property (1): By construction, Z is chosen uniformly from $\{1, 2, \ldots, M\}$ independently of the pair (X, Y). Therefore, $P_{XYZ} = P_{XY} \cdot P_Z$, which establishes that Z is independent of (X, Y).

Verification of Property (2): For each fixed value $z \in \mathcal{Z}$, we define $f_z(x)$ to be the most likely value of Y given $X = x$ under a randomly chosen quantization indexed by z. More precisely, using a random binning argument, we assign each pair (x, y) to one of M bins. Conditioned on $Z = z$, we set $f_z(x) = \arg\max_{y:\text{bin}(x,y)=z} P_{Y|X}(y|x)$ with ties broken arbitrarily.

By the properties of random binning and the fact that M is sufficiently large, with high probability over the choice of the binning function (which is determined by Z), for each x, there exists a unique y in the bin corresponding to z that has significant probability mass. Thus, conditioned on $Z = z$, we can define $Y = f_z(X)$ as a deterministic function.

Verification of Property (3): We now bound $H(Y|Z)$. By the chain rule for entropy:

$$H(Y|Z) = H(Y, Z) - H(Z) = H(Y) + H(Z|Y) - H(Z) \leq H(Y)$$

since Z is independent of Y.

However, we need a tighter bound. Using the quantization scheme, we have:

$$H(Y|Z) \leq H(Y|X,Z) + I(X:Y|Z)$$

Since conditioned on $Z = z$, Y is a deterministic function of X, we have $H(Y|X,Z) = 0$.

For the mutual information term, by the independence of Z and (X,Y):

$$I(X:Y|Z) = I(X:Y)$$

By the properties of the random quantization and covering arguments from rate-distortion theory, the number of bins M required to achieve this is:

$$\log M \leq I(X:Y) + 2\log I(X:Y) + O(1)$$

Since Z is uniformly distributed over M values and independent of (X,Y), and using the fact that $H(Y|Z) \leq \log M$ by the quantization property:

$$H(Y|Z) \leq I(X:Y) + 2\log I(X:Y) + O(1)$$

This completes the proof of all three properties. □

H.7.4 Gemini

Proof. Let P_{XY} denote the joint distribution of X and Y, and let Q_Y denote the marginal distribution of Y. We construct the random variable Z and the functional dependence of Y on X and Z using the rejection sampling framework established in the Strong Functional Representation Lemma (also known as the Reverse Shannon Theorem).

Construction of Z: Let Z be a Poisson Point Process (PPP) on the space $\mathcal{Y} \times \mathbb{R}_+$ with intensity measure $\mu(dy, dt) = Q_Y(dy) \times dt$. We can represent Z as a countable collection of points $\{(Y_k, T_k)\}_{k\in\mathbb{N}}$, where the T_k are arrival times of a Poisson process with unit rate, and conditioned on the times, the values Y_k are independent and identically distributed according to Q_Y. By definition, the construction of Z depends only on the marginal Q_Y, and thus Z is independent of the specific realization of (X,Y). This satisfies condition (1).

Functional Representation (Condition 2): For any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, define the density ratio (or Radon-Nikodym derivative) as:

$$w(y|x) = \frac{P_{Y|X}(y|x)}{Q_Y(y)}.$$

For a fixed x and a realization $z = \{(y_k, t_k)\}_k$, we define the index $K(x,z)$ as:

$$K(x,z) = \arg\min_{k\in\mathbb{N}} \frac{t_k}{w(y_k|x)}.$$

We define the variable Y conditioned on $X = x$ and $Z = z$ to be $Y_{K(x,z)}$. Conditioned on $Z = z$, the value of Y is uniquely determined by x (almost surely, as the probability of two points achieving the exact same minimum is zero). Thus, Y is a deterministic function of X given Z, satisfying condition (2).

To verify that this construction preserves the correct conditional distribution $P_{Y|X}$, consider the set of transformed points $\mathcal{P}_x = \{(Y_k, T_k/w(Y_k|x))\}_k$. By the mapping theorem for Poisson processes, $\mathcal{P}_x$ is a PPP with intensity measure:

$$\mu_x(dy, dt') = Q_Y(dy) \cdot w(y|x)\, dt' = P_{Y|X}(y|x)\, dt'.$$

The projection of $\mathcal{P}_x$ onto the time axis is a Poisson process with rate $\int_{\mathcal{Y}} P_{Y|X}(y|x)\, dy = 1$. The first arrival time $T^* = \min_k \frac{T_k}{w(Y_k|x)}$ follows an exponential distribution with mean 1. The associated mark Y^* (which is our constructed Y) is distributed according to the normalized intensity function, which is

$P_{Y|X}(y|x)$. Thus, the marginal distribution of the constructed Y given X matches the original conditional distribution.

Entropy Bound (Condition 3): We bound the conditional entropy $H(Y|Z)$. Since Y is determined by the index K given Z, we have $H(Y|Z) \leq H(K|Z)$. The index K depends on X. Let $D(x) = D_{KL}(P_{Y|X=x} \| Q_Y)$. It is a known result in the analysis of relative entropy rejection sampling (e.g., Harsha et al., 2010) that the expected code length (and thus the entropy) of the index K required to sample from $P_{Y|X=x}$ using samples from Q_Y is bounded by:

$$H(K|X = x, Z) \leq D(x) + 2\log(D(x) + 1) + O(1).$$

However, since K is a deterministic function of X and Z, $H(K|X, Z) = 0$. The relevant quantity is the entropy of K averaged over X given Z. Consider the communication cost interpretation. The index K can be encoded using a prefix-free code. The expected length of the index K (averaged over X and the randomness of Z) is bounded by the mutual information plus lower-order terms. Specifically, taking the expectation over X:

$$\mathbb{E}_X[D(P_{Y|X} \| Q_Y)] = I(X : Y).$$

By Jensen's inequality and the concavity of the logarithm, the bound on the entropy of the index K (which bounds $H(Y|Z)$) is:

$$H(Y|Z) \leq I(X : Y) + 2\log(I(X : Y) + 1) + O(1).$$

This satisfies condition (3). □

H.8 Problem 7.8

H.8.1 Problem Statement

Theorem H.8. *For every $\epsilon > 0$, one can simulate π with a protocol of length $O\left(\frac{\sqrt{IC}\log(C/\epsilon)}{\epsilon}\right)$. This simulation is internal and has error ϵ.*

H.8.2 GPT

Proof. Let $\epsilon > 0$ be given. We aim to construct a protocol that simulates π with error at most ϵ, and whose length is of order $O\left(\frac{\sqrt{IC}\log(C/\epsilon)}{\epsilon}\right)$.

We begin by noting that the simulation of π involves generating a sequence of random variables that approximate the value of π within the desired error bound ϵ. The protocol will use the following strategy.

1. Error Bound Setup: The simulation will be constructed in such a way that the total error incurred by the simulation at each step is bounded by ϵ. We aim to ensure that the cumulative error over all steps does not exceed ϵ.

2. Protocol Length Analysis: The protocol length is determined by the number of steps required to achieve the desired error bound. Each step involves a computation based on the current approximation of π, and the number of steps is proportional to the inverse of the error bound ϵ. Specifically, it can be shown that the length of the protocol is proportional to $\frac{\log(C/\epsilon)}{\epsilon}$, where C is some constant that represents the upper bound of the error term at each stage of the simulation. Additionally, the protocol will involve a factor of $\sqrt{IC}$, where IC represents the information complexity of the simulation.

3. Error Analysis: To guarantee that the simulation error remains within the desired bound ϵ, we use a technique based on controlling the growth of the error at each step. The error at each step is bounded by a term that decays with each successive computation, and the total accumulated error after a sufficient number of steps is guaranteed to be less than or equal to ϵ.

4. Final Protocol Construction: Combining the error bound and the protocol length analysis, we conclude that the simulation protocol has a length of $O\left(\frac{\sqrt{IC}\log(C/\epsilon)}{\epsilon}\right)$, and the total error in the simulation is at most ϵ.

Thus, the desired result is achieved, and the theorem is proven. □

H.8.3 Claude

Proof. Let π be a protocol with information cost IC and communication cost C. We construct an internal simulation of π with error at most ϵ and communication cost $O\left(\frac{\sqrt{IC}\log(C/\epsilon)}{\epsilon}\right)$.

The proof proceeds by dividing the protocol into rounds and using a sampling-based simulation technique.

First, we partition the protocol π into r rounds, where $r = \lceil C/\ell \rceil$ for a parameter ℓ to be determined. Each round consists of at most ℓ bits of communication from the original protocol.

For each round $i \in \{1, 2, \ldots, r\}$, we simulate the communication using a shared randomness technique. Let μ_i denote the distribution of transcripts up to round i in the original protocol π, conditioned on the inputs and prior randomness.

The key observation is that the information revealed in each round is bounded. Specifically, if T_i denotes the transcript up to round i, then by the chain rule for mutual information:

$$I(X, Y; T_i \mid T_{i-1}) \leq IC/r + \delta_i$$

where δ_i represents the information revealed in round i and the total information is bounded by IC.

For the simulation, in each round i, the players use shared randomness to sample from an approximation $\widehat{\mu}_i$ of the true distribution μ_i. By the sampling lemma, to achieve statistical distance at most ϵ/r between $\widehat{\mu}_i$ and μ_i, it suffices to use $O\left(\frac{1}{\epsilon^2}\log(1/\epsilon)\right)$ samples when the support size is appropriately bounded.

However, we employ a more refined approach using the information-theoretic structure. We use the fact that the protocol can be simulated using correlated sampling with communication proportional to the square root of the information cost per round.

Specifically, for each round, the communication required is:

$$O\left(\sqrt{\frac{IC}{r}} \cdot \log(r/\epsilon)\right)$$

Setting $r = \lceil C/\ell \rceil$ and optimizing over ℓ, the total communication becomes:

$$\sum_{i=1}^{r} O\left(\sqrt{\frac{IC}{r}} \cdot \log(r/\epsilon)\right) = O\left(r \cdot \sqrt{\frac{IC}{r}} \cdot \log(r/\epsilon)\right) = O\left(\sqrt{r \cdot IC} \cdot \log(r/\epsilon)\right)$$

To minimize this expression subject to $r \geq C/\ell$, we choose $r = O(C/\epsilon)$, which gives:

$$O\left(\sqrt{\frac{C}{\epsilon} \cdot IC} \cdot \log(C/\epsilon)\right) = O\left(\frac{\sqrt{IC \cdot C}}{\sqrt{\epsilon}} \cdot \log(C/\epsilon)\right)$$

Since we require the simulation to work for protocols with communication cost C, and using the fact that $C \geq IC$ (communication cost is at least information cost), we can refine the bound. By choosing the number of rounds appropriately as $r = \Theta(C/\epsilon)$ and applying the sampling technique with precision parameter ϵ/C, we obtain:

$$O\left(\frac{\sqrt{IC} \cdot \log(C/\epsilon)}{\epsilon}\right)$$

The error analysis follows from the union bound over all r rounds, where each round contributes error at most ϵ/r, giving total error at most ϵ.

Finally, the simulation is internal because at each step, both players can compute the same approximation $\widehat{\mu}_i$ using their shared randomness and the protocol structure, without revealing additional information beyond what is necessary for the simulation.

Therefore, the protocol π can be simulated with communication cost $O\left(\frac{\sqrt{IC}\log(C/\epsilon)}{\epsilon}\right)$ and error at most ϵ. □

H.8.4 Gemini

Proof. Let π be a protocol of depth C (the number of rounds). Let X and Y denote the inputs of Alice and Bob, respectively. The protocol π generates a transcript $\tau = (M_1, M_2, \ldots, M_C)$, where each message M_i depends on the inputs and the previous messages.

We define the Information Complexity (IC) of π as the internal information cost:

$$IC(\pi) = I(\tau; X|Y) + I(\tau; Y|X).$$

This can be expressed as the sum of the divergences between the conditional distributions of the messages and the priors. Specifically, for each round i, let P_i be the true distribution of the i-th message M_i given the history $\tau_{<i}$ and inputs. Let Q_i^A be Alice's view of the distribution of M_i (conditioned on X and $\tau_{<i}$) and Q_i^B be Bob's view. The information cost is bounded by the sum of the expected Kullback-Leibler divergences:

$$IC(\pi) \approx \sum_{i=1}^{C} \mathbb{E}_{\tau_{<i}} \left[D_{KL}(Q_i^A \| Q_i^B) + D_{KL}(Q_i^B \| Q_i^A) \right].$$

Let $D_i = \mathbb{E}[D_{KL}(Q_i^A \| Q_i^B) + D_{KL}(Q_i^B \| Q_i^A)]$ denote the expected divergence at round i. Thus, $IC = \sum_{i=1}^{C} D_i$.

To simulate π, Alice and Bob use a correlated sampling strategy. They share a source of public randomness. For each round $i = 1, \ldots, C$, they attempt to sample M_i using the shared randomness without communication. Let M_i^A and M_i^B be the samples generated by Alice and Bob respectively. By the Correlated Sampling Lemma (e.g., using Broder's or Holenstein's strategy), there exists a sampling procedure such that the probability of disagreement at round i is bounded by the statistical distance (Total Variation distance) between their distributions:

$$\Pr[M_i^A \neq M_i^B] \leq 2 \cdot TV(Q_i^A, Q_i^B).$$

Using Pinsker's inequality, $TV(P, Q) \leq \sqrt{\frac{1}{2} D_{KL}(P \| Q)}$. Therefore, the probability of error at round i, denoted δ_i, satisfies:

$$\delta_i \leq \sqrt{2D_i}.$$

The expected total number of disagreements K over the entire protocol is:

$$\mathbb{E}[K] = \sum_{i=1}^{C} \delta_i \leq \sum_{i=1}^{C} \sqrt{2D_i}.$$

By the Cauchy-Schwarz inequality, we bound the sum of square roots:

$$\sum_{i=1}^{C} \sqrt{D_i} \leq \sqrt{C \cdot \sum_{i=1}^{C} D_i} = \sqrt{C \cdot IC}.$$

Thus, $\mathbb{E}[K] \leq \sqrt{2C \cdot IC}$.

The simulation protocol proceeds as follows: 1. Alice and Bob generate the full candidate transcripts τ^A and τ^B using the shared randomness. 2. They engage in a reconciliation phase to find and correct disagreements. To do this efficiently, they can use a binary search or hash-based consistency checks on the path. 3. To ensure the simulation is correct with probability $1 - \epsilon$ (or to handle the error parameter), they use hash functions of length $O(\log(C/\epsilon))$ to detect mismatches. 4. For every mismatch found, they transmit the correct value. The cost to locate and correct one mismatch in a tree of depth C is $O(\log C + \log(1/\epsilon)) = O(\log(C/\epsilon))$.

The total expected communication cost L_{avg} is proportional to the expected number of mismatches times the cost per mismatch:

$$L_{avg} = O\left(\mathbb{E}[K] \cdot \log(C/\epsilon)\right) = O\left(\sqrt{C \cdot IC} \cdot \log(C/\epsilon)\right).$$

To obtain a protocol with a hard bound on the length that succeeds with error ϵ, we apply Markov's inequality. Let L_{max} be the maximum allowed length. The probability that the required communication exceeds L_{max} is:

$$\Pr[\text{Comm} > L_{max}] \leq \frac{L_{avg}}{L_{max}}.$$

We set this probability to be at most ϵ. Thus, we require:

$$L_{max} = \frac{L_{avg}}{\epsilon} = O\left(\frac{\sqrt{C \cdot IC}\log(C/\epsilon)}{\epsilon}\right).$$

If the communication exceeds L_{max}, the parties abort, contributing to the error ϵ. The simulation is internal (parties possess the transcript) and the total error is dominated by the abort probability ϵ (plus negligible hashing collision probabilities).

Thus, the protocol π can be simulated with length

$$O\left(\frac{\sqrt{IC \cdot C}\log(C/\epsilon)}{\epsilon}\right).$$

□

I Problems in Chapter 8

I.1 Problem 8.1

I.1.1 Problem Statement

Theorem I.1. *The decision tree complexity of $f : \{0,1\}^n \to \{0,1\}$ is at most n.*

I.1.2 GPT

Proof. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let the decision tree complexity of f be denoted by $D(f)$. The goal is to show that $D(f) \leq n$.

A decision tree is a binary tree where each internal node represents a test on one of the input bits of the function f, and each leaf corresponds to a particular output value of f. The depth of the decision tree is the maximum number of tests required to reach a leaf. The decision tree complexity $D(f)$ is the minimum depth of any decision tree that computes the function f.

First, note that the domain of f is the set of all binary strings of length n, i.e., $\{0,1\}^n$. Therefore, there are 2^n possible inputs to the function. For each input, the decision tree must output either 0 or 1 as the value of f on that input.

To determine the decision tree complexity, we observe that each internal node of the tree corresponds to testing a specific bit of the input string. In the worst case, the decision tree must test each of the n bits of the input to determine the value of f. Thus, the decision tree must have depth at most n to distinguish between all 2^n possible inputs.

We now argue that any decision tree for f must have depth at least n in the worst case. This follows from the fact that there are 2^n distinct input strings, and each leaf in the decision tree corresponds to one of these inputs. Since the decision tree is binary, the maximum number of leaves in a tree of depth d is 2^d. To distinguish between all 2^n possible inputs, the tree must have at least 2^n leaves. Therefore, the depth of the tree must be at least n.

Since the depth of any decision tree computing f is at least n and at most n, we conclude that the decision tree complexity of f is exactly n.

Thus, we have shown that the decision tree complexity of f is at most n. □

I.1.3 Claude

Proof. Let $f : \{0,1\}^n \to \{0,1\}$ be an arbitrary Boolean function. We will construct a decision tree that computes f using at most n queries.

Consider the following deterministic decision tree algorithm: Query the variables $x_1, x_2, \ldots, x_n$ in order. After querying all n variables, we have complete knowledge of the input $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$. At this point, we can compute $f(x)$ exactly and output the correct value.

More formally, we construct a complete binary tree of depth n where:

- At depth i (for $i = 0, 1, \ldots, n-1$), each internal node queries variable x_{i+1}.
- The left child corresponds to $x_{i+1} = 0$ and the right child corresponds to $x_{i+1} = 1$.
- Each leaf at depth n corresponds to a unique assignment $(a_1, a_2, \ldots, a_n) \in \{0,1\}^n$.
- Each leaf is labeled with the value $f(a_1, a_2, \ldots, a_n)$.

This decision tree has depth exactly n, and therefore uses at most n queries on any input. For any input $x \in \{0,1\}^n$, the algorithm follows a path from the root to a unique leaf determined by the values of $x_1, \ldots, x_n$, and outputs $f(x)$ correctly.

Since this construction works for any Boolean function $f : \{0,1\}^n \to \{0,1\}$, we conclude that the decision tree complexity of f is at most n. □

I.1.4 Gemini

Proof. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. The decision tree complexity of f, denoted by $D(f)$, is defined as the minimum depth of a deterministic decision tree that computes f. To prove that $D(f) \leq n$, it suffices to demonstrate the existence of a decision tree T that computes f and has depth at most n.

We construct a decision tree T as follows. The tree will be a complete binary tree of depth n. We associate the root of the tree with the variable x_1. For any internal node at depth k (where $0 \leq k < n$), we associate the variable x_{k+1}. Specifically, if the computation reaches a node at depth k, the tree queries the value of x_{k+1}. If $x_{k+1} = 0$, the computation proceeds to the left child; if $x_{k+1} = 1$, it proceeds to the right child.

This recursive construction ensures that every path from the root to a leaf has length exactly n. Each distinct path from the root to a leaf corresponds to a unique assignment of the input vector $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$. Let v be a leaf node reached by the path corresponding to the input x. We label the leaf v with the value $f(x)$.

For any input $x \in \{0,1\}^n$, the execution of the decision tree T queries $x_1, x_2, \ldots, x_n$ in order. The path taken is uniquely determined by the bits of x, leading to the specific leaf labeled with $f(x)$. Therefore, T correctly computes $f(x)$ for all inputs x.

The depth of the constructed tree T is n, as the maximum number of queries made on any input is n. Since $D(f)$ is the minimum depth over all valid decision trees for f, we conclude that

$$D(f) \leq \text{depth}(T) = n.$$

This completes the proof. □

I.1.5 Grok

Proof. Consider the decision tree where, at depth i for $0 \leq i < n$, the algorithm queries the bit x_{i+1}, regardless of previous answers. After querying all n bits, the leaf corresponding to the input string $x = (x_1, \ldots, x_n)$ outputs $f(x)$. This is a valid decision tree that computes f and has depth n. Therefore, the decision tree complexity of f is at most n. □

I.2 Problem 8.2

I.2.1 Problem Statement

Theorem I.2. *The function* $z_1 + z_2 + \cdots + z_n \pmod 2$ *has decision tree complexity* n.

I.2.2 GPT

Proof. Let the function be defined as $f(z_1, z_2, \ldots, z_n) = z_1 + z_2 + \cdots + z_n \mod 2$, where $z_1, z_2, \ldots, z_n \in \{0, 1\}$. The task is to show that the decision tree complexity of this function is n.

First, observe that the function $f(z_1, z_2, \ldots, z_n)$ is the sum of the input bits modulo 2. This is the parity of the input bits, which is known to be an example of an XOR function.

The decision tree complexity of a function refers to the minimum number of queries (or comparisons) that need to be made to evaluate the function on a given input. Specifically, for a decision tree computing the function, each query corresponds to testing the value of one of the input bits, and the goal is to determine the value of the function by asking as few queries as possible.

Consider the decision tree for the function $f(z_1, z_2, \ldots, z_n)$. Each query corresponds to testing the value of one bit z_i, for some $i \in \{1, 2, \ldots, n\}$. At each step in the decision tree, we query one of the bits and use its value to decide the next query. The tree branches depending on whether the queried bit is 0 or 1.

Since $f(z_1, z_2, \ldots, z_n)$ depends on all n input bits, the decision tree must query each bit at least once to determine the output. This is because the sum modulo 2 is sensitive to every bit in the input. If any bit is omitted, the parity may be incorrect, thus the function cannot be evaluated without querying all n bits.

To show that the decision tree complexity is exactly n, we argue that any decision tree computing this function must have at least n queries. This is because any decision tree computing $f(z_1, z_2, \ldots, z_n)$ must distinguish between all 2^n possible input combinations, as each combination corresponds to a different parity outcome. The number of leaves in the decision tree must therefore be at least 2^n, and since each query can distinguish between two branches, the depth of the tree (i.e., the number of queries) must be at least n to accommodate all possible input combinations.

Thus, the decision tree complexity of the function $f(z_1, z_2, \ldots, z_n)$ is exactly n, as any decision tree computing the function requires exactly n queries, and no decision tree can solve the problem with fewer queries.

Therefore, the decision tree complexity of $f(z_1, z_2, \ldots, z_n)$ is n. □

I.2.3 Claude

Proof. Let $f(z_1, z_2, \ldots, z_n) = z_1 + z_2 + \cdots + z_n \pmod 2$ where $z_i \in \{0, 1\}$ for all $i \in \{1, 2, \ldots, n\}$. We denote the decision tree complexity of f by $D(f)$. We will prove that $D(f) = n$ by establishing both a lower bound and an upper bound.

Upper Bound: We first show that $D(f) \leq n$. Consider the following deterministic decision tree: query the variables in order $z_1, z_2, \ldots, z_n$, and compute the running sum modulo 2. After querying all n variables, we can determine $f(z_1, \ldots, z_n)$ exactly. Therefore, there exists a decision tree of depth n that computes f, which gives us $D(f) \leq n$.

Lower Bound: We now show that $D(f) \geq n$. We proceed by contradiction. Suppose there exists a decision tree T computing f with depth strictly less than n. Then there exists at least one leaf ℓ in T that is reached by some path of length at most $n - 1$. This means that along the path to ℓ, at most $n - 1$ variables are queried.

Without loss of generality, suppose the variables queried along the path to ℓ are $z_{i_1}, z_{i_2}, \ldots, z_{i_k}$ where $k \leq n - 1$. Then there exists at least one variable, say z_j, that is not queried along this path.

Let $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ be an input that reaches leaf ℓ, where $a_i \in \{0, 1\}$ for all i. Since T computes f, the leaf ℓ must output the value $f(\mathbf{a}) = a_1 + a_2 + \cdots + a_n \pmod 2$.

Now consider the input $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ where $b_i = a_i$ for all $i \neq j$ and $b_j = 1 - a_j$. Since z_j is not queried along the path to ℓ, and $\mathbf{a}$ and $\mathbf{b}$ agree on all queried variables along this path, the input $\mathbf{b}$ must also reach the same leaf ℓ.

However, we have:

$$f(\mathbf{b}) = b_1 + b_2 + \cdots + b_n \pmod 2 = (a_1 + \cdots + a_{j-1} + (1 - a_j) + a_{j+1} + \cdots + a_n) \pmod 2$$

$$= (a_1 + \cdots + a_n + 1) \pmod 2 = f(\mathbf{a}) + 1 \pmod 2 \neq f(\mathbf{a})$$

This means that leaf ℓ must output both $f(\mathbf{a})$ and $f(\mathbf{b})$, which are distinct values. This is a contradiction, since each leaf can output only one value.

Therefore, every leaf in any decision tree computing f must be at depth at least n, which implies $D(f) \geq n$.

Combining the upper and lower bounds, we conclude that $D(f) = n$. □

I.2.4 Gemini

Proof. Let $f : \{0,1\}^n \to \{0,1\}$ be the function defined by $f(z_1, \ldots, z_n) = \sum_{i=1}^n z_i \pmod 2$. The decision tree complexity of f, denoted by $D(f)$, is the minimum depth of a deterministic decision tree that computes f. We aim to show that $D(f) = n$.

First, we establish the upper bound $D(f) \leq n$. Consider a decision tree T that queries the variables $z_1, z_2, \ldots, z_n$ in sequential order. At depth n, the path from the root to any leaf corresponds to a complete assignment of the input variables. The leaf is labeled with the value $\sum_{i=1}^n z_i \pmod 2$. Since this tree computes f correctly on all inputs and has depth n, we have $D(f) \leq n$.

Next, we establish the lower bound $D(f) \geq n$. Suppose, for the sake of contradiction, that there exists a decision tree T that computes f with depth $d < n$. By the definition of depth, the length of the longest path from the root to a leaf in T is d. Let L be a leaf node in T, and let P be the path from the root to L. Let $Q \subseteq \{1, \ldots, n\}$ be the set of indices of the variables queried along the path P. Since the length of P is at most d and $d < n$, we have $|Q| \leq d < n$. Therefore, the set of unqueried variables $\{1, \ldots, n\} \setminus Q$ is non-empty. Let $k \in \{1, \ldots, n\} \setminus Q$ be the index of a variable not queried on the path to L.

The path P defines a partial assignment to the variables in Q. Let $x \in \{0,1\}^n$ be an input consistent with this partial assignment, meaning that for all $j \in Q$, x_j matches the outcome of the query on the path. The decision tree T directs input x to leaf L, so the output of the tree is $T(x) = \text{label}(L)$.

Now, consider an input $x' \in \{0,1\}^n$ defined by flipping the k-th bit of x. That is, $x'_k = 1 - x_k$, and $x'_i = x_i$ for all $i \neq k$. Since $k \notin Q$, the variable z_k is not queried on the path P. Consequently, for all queried indices $j \in Q$, we have $x'_j = x_j$. This implies that x' is also consistent with the partial assignment defined by P, and the decision tree directs x' to the same leaf L. Thus, the tree produces the same output for both inputs: $T(x') = T(x)$.

However, we must evaluate the function f on these inputs. We have:

$$f(x') = \sum_{i=1}^n x'_i \pmod 2 = \left(\sum_{i \neq k} x_i + (1 - x_k) \right) \pmod 2.$$

This simplifies to:

$$f(x') = \left(\sum_{i=1}^n x_i + 1 - 2x_k \right) \pmod 2 \equiv f(x) + 1 \pmod 2.$$

Since $f(x) + 1 \not\equiv f(x) \pmod 2$, it follows that $f(x') \neq f(x)$. This contradicts the fact that $T(x') = T(x)$, as T is assumed to compute f correctly.

The contradiction implies that our assumption $d < n$ was false. Therefore, any decision tree computing f must query every variable on every path from the root to a leaf. It follows that the depth of any such tree must be at least n, so $D(f) \geq n$.

Combining the upper and lower bounds, we conclude that $D(f) = n$. □

I.2.5 Grok

Proof. The decision tree complexity of a boolean function $f : \{0,1\}^n \to \{0,1\}$ is the minimal depth over all decision trees computing f, where the depth is the length of the longest root-to-leaf path, and each internal node queries one variable z_i.

Here, $f(z_1, \ldots, z_n) = z_1 + \cdots + z_n \pmod 2$.

First, we establish the upper bound. Consider a decision tree that queries $z_1, z_2, \ldots, z_n$ in sequence along every path. At each leaf, the parity can be computed from the answers to all queries. This tree has depth n and computes f, so the complexity is at most n.

Next, we prove the lower bound by contradiction. Suppose there exists a decision tree T computing f with depth at most $n-1$. Then, every root-to-leaf path in T queries at most $n-1$ variables.

Consider an arbitrary leaf ℓ in T, reached via a path P that queries a set $S \subseteq \{1, \ldots, n\}$ of variables, with $|S| \leq n-1$. Thus, there exists some index $j \notin S$. The output at ℓ is a fixed bit $b \in \{0,1\}$.

Let σ denote the fixed answers to the queries along P (i.e., the values of the variables in S). Consider two inputs $x, y \in \{0,1\}^n$ such that: - $x_i = y_i = \sigma(i)$ for all $i \in S$, - $x_j = 0$ and $y_j = 1$, - $x_k = y_k = 0$ for all $k \notin S \cup \{j\}$.

Both x and y are consistent with the answers along P, so both reach ℓ and receive output b. However, x and y differ only in the j-th coordinate, so

$$f(y) = f(x) + 1 \pmod 2.$$

This implies $b = b + 1 \pmod 2$, a contradiction.

Therefore, no such decision tree T of depth at most $n-1$ exists, so the complexity is at least n.

Hence, the decision tree complexity of f is exactly n. □

I.3 Problem 8.3

I.3.1 Problem Statement

Theorem I.3. *Show that there is a decision tree with $O(n)$ vertices that computes a function $f\colon \{0,1\}^n \to \{0,1\}$, so that the decision tree complexity of f is n. Conclude that decision trees cannot be balanced.*

I.3.2 GPT

Proof. Define $f : \{0,1\}^n \to \{0,1\}$ by $f(x_1, \ldots, x_n) = x_1$. Consider the following decision tree T: the root queries the bit x_1; it has two outgoing edges labeled 0 and 1, leading to two leaves labeled 0 and 1, respectively. Then T computes f because on input x it queries x_1 and outputs the same value.

The number of vertices of T is 3, hence $|V(T)| = O(n)$.

We now prove that the decision tree complexity of f is n. Let D be any deterministic decision tree that computes f. We claim that for every input $x \in \{0,1\}^n$, along the computation path of D on x the variable x_1 must be queried at some node. Suppose for contradiction that there exists an input x whose computation path in D queries no occurrence of x_1. Let $y \in \{0,1\}^n$ be defined by $y_1 = 1 - x_1$ and $y_i = x_i$ for all $2 \leq i \leq n$. Along the computation path of D on x, the queried variables are among $\{x_2, \ldots, x_n\}$, and x and y agree on all these queried bits. Therefore D follows the same computation path on y as on x and outputs the same value on both inputs. Since D computes f, this implies $f(x) = f(y)$. But $f(x) = x_1$ and $f(y) = y_1 = 1 - x_1$, hence $f(x) \neq f(y)$, a contradiction. Thus x_1 is queried on every computation path, and consequently the depth of D is at least 1. Conversely, the tree T above computes f with depth 1, so the decision tree complexity of f equals 1.

To obtain decision tree complexity n with $O(n)$ vertices, instead define $g : \{0,1\}^n \to \{0,1\}$ by $g(x_1, \ldots, x_n) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$, where $\oplus$ denotes addition modulo 2. Consider the decision tree T_g that queries x_1 at the root, then queries x_2 on the next node along the unique path, and so on, querying x_i at depth $i-1$ for each $1 \leq i \leq n$, and finally outputs the parity of the queried bits at the unique leaf reached. Formally, T_g is a path of n internal vertices labeled $x_1, \ldots, x_n$ in order, with two outgoing edges at each internal vertex labeled 0 and 1, and with 2^n leaves labeled by the parity of the sequence of edge labels along the path. The tree T_g computes g by construction, and it has n internal vertices, hence $O(n)$ internal vertices. Since the theorem asks for $O(n)$ vertices, we instead compress the leaves: replace the 2^n leaves by a single final internal vertex that computes parity of the answers and has two leaves. This is not allowed in the standard decision tree model, so we modify the construction as follows.

Define $h : \{0,1\}^n \to \{0,1\}$ by

$$h(x_1, \ldots, x_n) = \begin{cases} 0 & \text{if } (x_1, \ldots, x_{n-1}) \neq (1, \ldots, 1), \\ x_n & \text{if } (x_1, \ldots, x_{n-1}) = (1, \ldots, 1). \end{cases}$$

We build a decision tree T_h as a single path: at depth 0 query x_1; if the answer is 0, go to a leaf labeled 0; if the answer is 1, proceed to query x_2; if the answer is 0, go to a leaf labeled 0; otherwise proceed, and so

on. After querying x_{n-1} and receiving answer 1, query x_n and output its value at two leaves labeled 0 and 1. This tree has exactly one internal vertex for each queried variable along the path, hence n internal vertices, and it has one leaf for each early termination plus two final leaves, hence $(n-1)+2 = n+1$ leaves. Therefore $|V(T_h)| = n + (n+1) = 2n+1 = O(n)$. By construction, T_h computes h.

We now prove that the decision tree complexity of h is n. Let D be any deterministic decision tree computing h. Consider the input $a = (1, 1, \ldots, 1, 0) \in \{0,1\}^n$ and the input $b = (1, 1, \ldots, 1, 1) \in \{0,1\}^n$, which differ only in the last coordinate. We have $h(a) = 0$ and $h(b) = 1$, since $(a_1, \ldots, a_{n-1}) = (1, \ldots, 1) = (b_1, \ldots, b_{n-1})$ and the output equals the last bit. Hence D must output different values on a and b, so the computation paths of D on a and b cannot terminate at the same leaf. Since a and b agree on the first $n-1$ coordinates, the only way for D to distinguish them is to query x_n on at least one of these two inputs, and in fact it must query x_n on both inputs along their computation paths; otherwise, if x_n is not queried on, say, a, then the sequence of answers to all queried variables is identical on a and b (because all queried variables would lie in $\{x_1, \ldots, x_{n-1}\}$), forcing the same output on both, contradicting $h(a) \neq h(b)$. Thus x_n is queried on both computation paths.

Next, fix $k \in \{1, \ldots, n-1\}$ and consider the two inputs $u^{(k)}$ and $v^{(k)}$ defined by

$$u_i^{(k)} = \begin{cases} 1 & \text{if } i < k, \\ 0 & \text{if } i = k, \\ 0 & \text{if } i > k, \end{cases} \qquad v_i^{(k)} = \begin{cases} 1 & \text{if } i < k, \\ 1 & \text{if } i = k, \\ 0 & \text{if } i > k, \end{cases}$$

so that $u^{(k)}$ and $v^{(k)}$ agree on coordinates $1, \ldots, k-1$ and on coordinates $k+1, \ldots, n$, and differ only at coordinate k. For both inputs, the prefix $(x_1, \ldots, x_{n-1})$ is not $(1, \ldots, 1)$, because in $u^{(k)}$ we have $x_k = 0$, and in $v^{(k)}$ we have $x_{k+1} = 0$ (since $k \leq n-1$ implies $k+1 \leq n$ and by definition $v_{k+1}^{(k)} = 0$). Therefore $h(u^{(k)}) = 0$ and $h(v^{(k)}) = 0$, so this pair does not force a query. Instead, consider the inputs

$$p_i^{(k)} = \begin{cases} 1 & \text{if } i \neq k, \\ 0 & \text{if } i = k, \end{cases} \qquad q_i^{(k)} = \begin{cases} 1 & \text{if } 1 \leq i \leq n, \end{cases}$$

and additionally set $p_n^{(k)} = 0$ and $q_n^{(k)} = 0$. Then $p^{(k)}$ and $q^{(k)}$ differ only at coordinate k, they satisfy $(p_1^{(k)}, \ldots, p_{n-1}^{(k)}) \neq (1, \ldots, 1)$ while $(q_1^{(k)}, \ldots, q_{n-1}^{(k)}) = (1, \ldots, 1)$, and both have last bit 0. Hence $h(p^{(k)}) = 0$ by the first case in the definition, while $h(q^{(k)}) = q_n^{(k)} = 0$ by the second case, so again no separation occurs. To force separation, set $p_n^{(k)} = 0$ and $q_n^{(k)} = 1$ while keeping other coordinates as above, but then they differ in two coordinates. We therefore proceed differently.

Let $S = \{(1, \ldots, 1, t) : t \in \{0,1\}\}$ be the set of two inputs with the first $n-1$ bits all equal to 1. On S, the function h coincides with the projection $x \mapsto x_n$. As shown above, any decision tree computing h must query x_n on both inputs in S. Moreover, on the input $(1, \ldots, 1, 0)$, the decision tree must not terminate before knowing that $(x_1, \ldots, x_{n-1}) = (1, \ldots, 1)$, because if it terminates at a leaf after having queried a strict subset of $\{x_1, \ldots, x_{n-1}\}$ and having received all answers equal to 1, then the same transcript is consistent with an input z that matches those queried bits and has $z_n = 1$ and also has all unqueried bits among $x_1, \ldots, x_{n-1}$ equal to 1, namely $z = (1, \ldots, 1, 1)$, on which $h(z) = 1$, forcing an incorrect output. Formally, let the computation path of D on $(1, \ldots, 1, 0)$ query variables in some set $Q \subseteq \{1, \ldots, n\}$, and suppose there exists $j \in \{1, \ldots, n-1\} \setminus Q$. Then define z by $z_i = 1$ for $1 \leq i \leq n-1$, $z_n = 1$. Since D did not query x_j on the path, the answers to all queries on $(1, \ldots, 1, 0)$ and on z coincide for every queried variable other than possibly x_n. If x_n is not queried before termination, then the transcripts coincide entirely, forcing equal outputs, contradicting $h(1, \ldots, 1, 0) = 0 \neq 1 = h(z)$. If x_n is queried, then on $(1, \ldots, 1, 0)$ the answer is 0 while on z it is 1, so D can distinguish them. However, for correctness on inputs where $(x_1, \ldots, x_{n-1}) \neq (1, \ldots, 1)$, D must also correctly output 0 even when $x_n = 1$. In particular, consider the input w obtained by setting $w_i = 1$ for all $i \neq j$, $w_j = 0$, and $w_n = 1$. Then $(w_1, \ldots, w_{n-1}) \neq (1, \ldots, 1)$ so $h(w) = 0$. The transcript on $(1, \ldots, 1, 0)$ along the path has all queried bits among $x_1, \ldots, x_{n-1}$ equal to 1 and $x_n = 0$; the transcript on w matches all queried bits among $x_1, \ldots, x_{n-1}$ (since j is unqueried and all queried ones are 1) but has $x_n = 1$. If the path on $(1, \ldots, 1, 0)$

queries x_n, it would see different answers on these two inputs and could in principle output differently; but it must output 0 on both. Hence querying x_n alone does not certify the case distinction. To avoid a wrong output on $z = (1, \ldots, 1, 1)$, the tree must ensure that all of $x_1, \ldots, x_{n-1}$ are 1 before it can output x_n as the result. Therefore, on input $(1, \ldots, 1, 0)$, the computation must query every variable $x_1, \ldots, x_{n-1}$, and also query x_n to determine the final output. Thus the computation path length on this input is at least n, and consequently the worst-case depth of D is at least n.

On the other hand, the explicit tree T_h described above queries at most n variables on any input, hence has depth n. Therefore the decision tree complexity of h is exactly n.

Finally, T_h has $O(n)$ vertices and depth n. In any rooted binary tree of depth n whose internal nodes all have outdegree 2, the number of leaves is at most 2^n, with equality only for the complete binary tree; conversely, any balanced binary tree of depth n has $\Omega(2^n)$ vertices, since the number of vertices is at least the number of leaves. Since T_h has only $O(n)$ vertices while depth n, it cannot be balanced in the sense that its depth is $\Theta(\log |V(T_h)|)$; indeed $\log |V(T_h)| = \Theta(\log n)$ while the depth is n. Hence decision trees cannot in general be balanced without potentially increasing the number of vertices from $O(n)$ to exponential in n. □

I.3.3 Claude

Proof. We prove the theorem in two parts: first constructing a decision tree with $O(n)$ vertices that computes a function with decision tree complexity n, and then concluding that decision trees cannot always be balanced.

Part 1: Construction of the decision tree.

Define the function $f\colon \{0,1\}^n \to \{0,1\}$ by

$$f(x_1, x_2, \ldots, x_n) = x_1 \oplus x_2 \oplus \cdots \oplus x_n,$$

where $\oplus$ denotes the XOR operation (addition modulo 2).

We construct a decision tree T that computes f as follows. The tree queries the variables in order $x_1, x_2, \ldots, x_n$. At each internal node at depth i (for $i = 1, 2, \ldots, n-1$), we query variable x_i. The tree branches left if $x_i = 0$ and right if $x_i = 1$. After querying all n variables, we reach a leaf that outputs the XOR of all queried values.

This decision tree has exactly n internal nodes (one for each variable) and 2^n leaves. However, we can optimize this tree by observing that after querying $x_1, x_2, \ldots, x_{n-1}$, the value of f is completely determined once we query x_n. More precisely, we construct a linear decision tree: query x_1, then x_2, then x_3, and so on, up to x_n. This gives us a path of length n with n internal nodes and 2 leaves at the bottom.

The total number of vertices in this tree is $n + 2 = O(n)$.

Verification that the decision tree complexity is n:

To show that the decision tree complexity of f is exactly n, we must prove that any decision tree computing f requires depth at least n.

Suppose, for contradiction, that there exists a decision tree T' computing f with depth less than n. Then there exists some input $x \in \{0,1\}^n$ such that T' computes $f(x)$ by querying fewer than n variables. Without loss of generality, suppose T' does not query variable x_j for some $j \in \{1, 2, \ldots, n\}$ on input x.

Let $x' \in \{0,1\}^n$ be identical to x except that $x'_j = 1 - x_j$ (i.e., we flip the j-th bit). Since T' does not query x_j on input x, and x and x' agree on all other coordinates, the decision tree T' follows the same path for both x and x', and thus outputs the same value: $T'(x) = T'(x')$.

However, we have

$$f(x') = x_1 \oplus \cdots \oplus x_{j-1} \oplus (1 - x_j) \oplus x_{j+1} \oplus \cdots \oplus x_n = f(x) \oplus 1 \neq f(x).$$

This contradicts the assumption that T' correctly computes f. Therefore, any decision tree computing f must have depth at least n, so the decision tree complexity of f is exactly n.

Part 2: Decision trees cannot be balanced.

A balanced binary tree with n internal nodes has depth $\Theta(\log n)$. However, we have constructed a function f with decision tree complexity n that can be computed by a decision tree with $O(n)$ vertices. Any decision tree computing f must have depth n, which is $\Theta(n)$.

If all decision trees could be balanced, then any function computable by a decision tree with n internal nodes would have decision tree complexity $O(\log n)$. But our function f has a decision tree with $O(n)$ vertices (specifically, n internal nodes) yet requires depth $n = \Theta(n)$, not $O(\log n)$.

Therefore, decision trees cannot always be balanced. □

I.3.4 Gemini

Proof. Let $f : \{0,1\}^n \to \{0,1\}$ be the logical AND function, defined by $f(x_1, \ldots, x_n) = \bigwedge_{i=1}^n x_i$. We first determine the decision tree complexity $D(f)$.

To show that $D(f) = n$, we must show that any decision tree computing f has depth at least n. Suppose, for the sake of contradiction, that there exists a decision tree T computing f with depth $d < n$. Consider the input $x = (1, 1, \ldots, 1)$. Since $f(x) = 1$, the computation path for x in T must end at a leaf labeled 1. Let P be the set of indices of the variables queried on this path. Since the length of the path is at most d, we have $|P| \leq d < n$. Consequently, there exists an index $j \in \{1, \ldots, n\}$ such that $j \notin P$.

Now, consider the input y defined such that $y_j = 0$ and $y_k = 1$ for all $k \neq j$. Since y differs from x only at index j, and index j is not queried on the path for x, the decision tree must follow the same path for y as it does for x. Therefore, T outputs 1 for input y. However, by the definition of the AND function, $f(y) = 0$. This contradicts the assumption that T computes f. Thus, any decision tree for f must have depth at least n. Since a trivial tree of depth n exists, we conclude $D(f) = n$.

Next, we demonstrate the existence of a decision tree for f with $O(n)$ vertices. Consider the tree T_{AND} constructed as follows:

- The root node queries x_1.
- For each $i \in \{1, \ldots, n-1\}$, the node querying x_i has an outgoing edge labeled 0 pointing to a leaf node with value 0, and an outgoing edge labeled 1 pointing to the node querying x_{i+1}.
- The node querying x_n has an outgoing edge labeled 0 pointing to a leaf node with value 0, and an outgoing edge labeled 1 pointing to a leaf node with value 1.

We count the vertices in T_{AND}. There are n internal nodes corresponding to the variables $x_1, \ldots, x_n$. For each of the first $n-1$ variables, there is a distinct leaf labeled 0. The last variable x_n connects to one leaf labeled 0 and one leaf labeled 1. Thus, the total number of leaves is $(n-1) + 2 = n + 1$. The total number of vertices is the sum of internal nodes and leaves:

$$|V(T_{AND})| = n + (n+1) = 2n + 1.$$

Since $2n + 1 \in O(n)$, we have constructed a decision tree with $O(n)$ vertices computing f.

Finally, we address the property of balance. A tree with V vertices is typically considered balanced if its depth is $O(\log V)$. For the function f, we have shown that the minimum depth required is n. For the tree T_{AND}, the size is $V = 2n + 1$. If T_{AND} were balanced, its depth would be proportional to $\log(2n + 1)$. However, the depth is n. Since n is not $O(\log n)$, this tree is not balanced. Furthermore, since any decision tree computing f requires depth n, and any such tree must have size at least n (to query n variables), the depth n will never be $O(\log V)$ for any polynomial size tree. Thus, decision trees cannot be balanced in general. □

I.3.5 Grok

Proof. Consider the function $f : \{0,1\}^n \to \{0,1\}$ defined by $f(x_1, \ldots, x_n) = 1$ if and only if $x_1 = x_2 = \cdots = x_n = 1$, and $f(x_1, \ldots, x_n) = 0$ otherwise. That is, f is the n-bit AND function.

We first construct a decision tree T that computes f and has $O(n)$ vertices. The tree T is structured as follows: The root queries x_1. If $x_1 = 0$, output 0; if $x_1 = 1$, proceed to a child node that queries x_2. At the node querying x_i (for $1 \leq i < n$), if $x_i = 0$, output 0; if $x_i = 1$, proceed to a child node that queries x_{i+1}. At the node querying x_n, if $x_n = 0$, output 0; if $x_n = 1$, output 1. This tree has n internal nodes (one for each variable) and $n + 1$ leaves (one leaf outputting 0 after each query branch on 0, and one leaf outputting 1 at the end), for a total of $2n + 1 = O(n)$ vertices. It is straightforward to verify that T computes f.

Next, we show that the decision tree complexity of f is n, meaning that any decision tree computing f has depth at least n. Suppose, for the sake of contradiction, that there exists a decision tree T' computing f with depth $d < n$. Then T' has at least one leaf ℓ reachable by a path of length at most d, so at least one variable x_j is not queried along this path. Let $b \in \{0, 1\}$ be the output at ℓ.

If $b = 1$, consider two inputs that follow this path: one where all unqueried variables (including x_j) are set to 1, and another where $x_j = 0$ and all other unqueried variables are set to 1. Both inputs reach ℓ and output 1. However, the second input has $x_j = 0$, so f should output 0, a contradiction.

Since the case $b = 1$ leads to a contradiction, all leaves at depth less than n must output 0. However, the input where all $x_i = 1$ must output 1, so its path in T' must reach a leaf outputting 1. But all such leaves would require querying all n variables (otherwise, the above contradiction applies), so the path to any 1-leaf has length n. Thus, the depth of T' is at least n, contradicting the assumption that $d < n$. Therefore, the decision tree complexity of f is n.

Since there exists a decision tree computing f with $O(n)$ vertices and depth n (the minimal possible depth), but any balanced binary tree of depth n (where all leaves are at depth n or $n - 1$) has $\Theta(2^n)$ vertices, it follows that decision trees computing Boolean functions cannot always be balanced. □

I.4 Problem 8.4

I.4.1 Problem Statement

Theorem I.4. *Let* $\mathsf{IND} : [d] \times \{0, 1\}^d \to \{0, 1\}$ *be the indexing function, defined by* $\mathsf{IND}(i, y) = y_i$*. There exists a constant* $1 > \epsilon > 0$ *such that for every* $b \in \{0, 1\}$*, there is a distribution on* b*-monochromatic rectangles* $R_b \subseteq [d] \times \{0, 1\}^d$ *such that for every* $U \subseteq [d]$ *and* $V \subseteq \{0, 1\}^d$ *with* $|U| \geq d^{1-\epsilon}$ *and* $|V| \geq 2^d/d^\epsilon$*,*

$$\Pr[R_b \cap (U \times V) \neq \emptyset] \geq 1/2.$$

I.4.2 GPT

Proof. We are given the indexing function IND : $[d] \times \{0, 1\}^d \to \{0, 1\}$, where $\text{IND}(i, y) = y_i$ for $i \in [d]$ and $y \in \{0, 1\}^d$. The goal is to show that for each $b \in \{0, 1\}$, there exists a distribution on b-monochromatic rectangles $R_b \subseteq [d] \times \{0, 1\}^d$ such that for every $U \subseteq [d]$ and $V \subseteq \{0, 1\}^d$ with $|U| \geq d^{1-\epsilon}$ and $|V| \geq d^{1/\epsilon}$, the probability $\Pr[R_b \cap (U \times V) \neq \emptyset] \geq \frac{1}{2}$ for some constant $1 > \epsilon > 0$.

Step 1: Definitions and Notations

Let $b \in \{0, 1\}$. A b-monochromatic rectangle R_b is defined as a subset of $[d] \times \{0, 1\}^d$ such that for all pairs $(i, y) \in R_b$, we have $y_i = b$. That is, R_b contains all points (i, y) where the i-th coordinate of y is fixed to be b.

We are asked to show that there exists a distribution on such rectangles R_b that satisfies the required condition for every subset $U \subseteq [d]$ and $V \subseteq \{0, 1\}^d$ with $|U| \geq d^{1-\epsilon}$ and $|V| \geq d^{1/\epsilon}$.

Step 2: Construction of the Distribution

We construct a distribution over the set of b-monochromatic rectangles. A natural choice is to consider a random rectangle R_b defined by fixing a subset $U' \subseteq [d]$ of size $|U'| \geq d^{1-\epsilon}$ and a subset $V' \subseteq \{0, 1\}^d$ of size $|V'| \geq d^{1/\epsilon}$. The rectangle R_b is chosen so that R_b contains all pairs (i, y) such that $i \in U'$ and $y \in V'$, and for each $i \in U'$, the value y_i is fixed to be b.

The probability distribution over such rectangles can be defined in a way that guarantees the required intersection condition with high probability.

Step 3: Estimation of the Probability

Let us now estimate the probability $\Pr[R_b \cap (U \times V) \neq \emptyset]$ where $U \subseteq [d]$ and $V \subseteq \{0, 1\}^d$ satisfy the conditions $|U| \geq d^{1-\epsilon}$ and $|V| \geq d^{1/\epsilon}$. The rectangle R_b intersects the set $U \times V$ if there is at least one point $(i, y) \in R_b$ such that $i \in U$ and $y \in V$.

Given that the rectangle R_b is defined by selecting a subset U' of U and a subset V' of V, the size of U' and V' is sufficiently large. Specifically, $|U'| \geq d^{1-\epsilon}$ and $|V'| \geq d^{1/\epsilon}$, which implies that the rectangle R_b covers a significant portion of the set $U \times V$.

Thus, the probability that R_b intersects $U \times V$ is at least $\frac{1}{2}$ when the conditions on $|U|$ and $|V|$ are satisfied. This follows from the fact that the distribution of R_b is designed to ensure that a substantial portion of $U \times V$ is covered by the random rectangle R_b.

Step 4: Conclusion

We have shown that for each $b \in \{0,1\}$, there exists a distribution over b-monochromatic rectangles R_b such that for every $U \subseteq [d]$ and $V \subseteq \{0,1\}^d$ with $|U| \geq d^{1-\epsilon}$ and $|V| \geq d^{1/\epsilon}$, the probability $\Pr[R_b \cap (U \times V) \neq \emptyset] \geq \frac{1}{2}$. This completes the proof. □

I.4.3 Claude

Proof. We construct a distribution on b-monochromatic rectangles for each $b \in \{0,1\}$ and prove the hitting property.

Fix $b \in \{0,1\}$. We define a distribution $\mathcal{D}_b$ on b-monochromatic rectangles as follows. Sample uniformly at random an index $i^* \in [d]$ and let $R_b = \{i^*\} \times V_b$ where $V_b = \{y \in \{0,1\}^d : y_{i^*} = b\}$. Note that R_b is indeed a b-monochromatic rectangle since $\mathsf{IND}(i^*, y) = y_{i^*} = b$ for all $(i^*, y) \in R_b$.

Let $\epsilon = 1/3$. We verify that this choice satisfies $0 < \epsilon < 1$.

Now consider arbitrary sets $U \subseteq [d]$ and $V \subseteq \{0,1\}^d$ with $|U| \geq d^{1-\epsilon} = d^{2/3}$ and $|V| \geq 2^d/d^\epsilon = 2^d/d^{1/3}$.

For a randomly chosen rectangle R_b according to $\mathcal{D}_b$, we have $R_b = \{i^*\} \times V_b$ where i^* is uniform in $[d]$ and $V_b = \{y \in \{0,1\}^d : y_{i^*} = b\}$.

The rectangle R_b intersects $U \times V$ if and only if $i^* \in U$ and $V_b \cap V \neq \emptyset$, which is equivalent to $i^* \in U$ and there exists $y \in V$ with $y_{i^*} = b$.

For each $i \in [d]$, define $V_i^b = \{y \in V : y_i = b\}$. Then:

$$\Pr[R_b \cap (U \times V) \neq \emptyset] = \Pr[i^* \in U \text{ and } V_{i^*}^b \neq \emptyset] = \frac{1}{d} \sum_{i \in U} \mathbf{1}[V_i^b \neq \emptyset]$$

For each $y \in V$, exactly one of $y_i = 0$ or $y_i = 1$ holds for each coordinate i. Therefore, for each $i \in [d]$:

$$|V_i^0| + |V_i^1| = |V|$$

Summing over all $i \in [d]$:

$$\sum_{i \in [d]} |V_i^b| \geq \frac{1}{2} \sum_{i \in [d]} (|V_i^0| + |V_i^1|) = \frac{d \cdot |V|}{2}$$

Since $|V_i^b| > 0$ implies $V_i^b \neq \emptyset$, and using the fact that $|V_i^b| \leq |V|$ for all i, we obtain:

$$\sum_{i \in [d]} \mathbf{1}[V_i^b \neq \emptyset] \geq \frac{\sum_{i \in [d]} |V_i^b|}{|V|} \geq \frac{d}{2}$$

Therefore, the number of indices $i \in U$ with $V_i^b \neq \emptyset$ is at least:

$$\sum_{i \in U} \mathbf{1}[V_i^b \neq \emptyset] \geq \sum_{i \in [d]} \mathbf{1}[V_i^b \neq \emptyset] - (d - |U|) \geq \frac{d}{2} - (d - d^{2/3})$$

For sufficiently large d, we have $d - d^{2/3} \leq d/4$, which gives:

$$\sum_{i \in U} \mathbf{1}[V_i^b \neq \emptyset] \geq \frac{d}{2} - \frac{d}{4} = \frac{d}{4}$$

Thus:

$$\Pr[R_b \cap (U \times V) \neq \emptyset] = \frac{1}{d} \sum_{i \in U} \mathbf{1}[V_i^b \neq \emptyset] \geq \frac{1}{d} \cdot \frac{d}{4} = \frac{1}{4}$$

By adjusting the constant ϵ appropriately (choosing ϵ slightly larger), we can ensure the probability is at least $1/2$ as required. □

I.4.4 Gemini

Proof. We show that there exists a constant $\epsilon > 0$ satisfying the theorem statement. Let $\epsilon = 1/5$. Consider any $b \in \{0, 1\}$. We define the distribution $\mathcal{D}_b$ on b-monochromatic rectangles as follows. Let $k = \lfloor d^{2\epsilon} \rfloor = \lfloor d^{2/5} \rfloor$. We choose a subset $J \subseteq [d]$ of size k uniformly at random. For a chosen J, we define the rectangle R_J by

$$R_J = J \times \{y \in \{0, 1\}^d \mid \forall j \in J, y_j = b\}.$$

First, we verify that R_J is a b-monochromatic rectangle. Let $(i, y) \in R_J$. By definition, $i \in J$ and $y_j = b$ for all $j \in J$. In particular, $y_i = b$. Thus, $\mathsf{IND}(i, y) = b$ for all $(i, y) \in R_J$.

Now, let $U \subseteq [d]$ and $V \subseteq \{0, 1\}^d$ be arbitrary sets satisfying $|U| \geq d^{1-\epsilon}$ and $|V| \geq 2^d/d^\epsilon$. We analyze the probability that $R_J \cap (U \times V) \neq \emptyset$. Note that

$$R_J \cap (U \times V) \neq \emptyset \iff (J \cap U \neq \emptyset) \wedge (\{y \mid y|_J = \vec{b}\} \cap V \neq \emptyset).$$

Let E_1 be the event that $J \cap U = \emptyset$, and let E_2 be the event that $\{y \mid y|_J = \vec{b}\} \cap V = \emptyset$. We want to show that $\Pr[\neg E_1 \wedge \neg E_2] \geq 1/2$, which holds if $\Pr[E_1] + \Pr[E_2] \leq 1/2$.

Bounding $\Pr[E_1]$: The set J is a random subset of size k. The probability that J misses U is bounded by

$$\Pr[E_1] = \frac{\binom{d-|U|}{k}}{\binom{d}{k}} \leq \left(1 - \frac{|U|}{d}\right)^k \leq \exp\left(-\frac{k|U|}{d}\right).$$

Substituting the bounds $|U| \geq d^{1-\epsilon}$ and $k = \lfloor d^{2\epsilon} \rfloor$:

$$\frac{k|U|}{d} \geq \frac{(d^{2\epsilon} - 1)d^{1-\epsilon}}{d} = d^\epsilon - d^{-\epsilon}.$$

For sufficiently large d, d^ϵ is large, so $\Pr[E_1] \leq \exp(-(d^{1/5} - o(1)))$, which approaches 0. Thus, for large d, $\Pr[E_1] \leq 1/4$.

Bounding $\Pr[E_2]$: Let $\mu = \frac{|V|}{2^d} \geq d^{-\epsilon}$. For a fixed J, let $C_J = \{y \in \{0, 1\}^d \mid y|_J = \vec{b}\}$ be the subcube corresponding to J. The size of C_J is 2^{d-k}. Let X_J be the random variable representing the density of V within this subcube:

$$X_J = \frac{|V \cap C_J|}{2^{d-k}}.$$

The event E_2 is equivalent to $X_J = 0$. We calculate the expectation and variance of X_J over the random choice of J. By symmetry, the expected density in a random subcube is equal to the global density:

$$\mathbb{E}[X_J] = \mu \geq d^{-\epsilon}.$$

For the variance, we utilize the standard bound for the density of a set under random restrictions. For a random set J of size k,

$$\text{Var}(X_J) \leq \frac{k}{d-k}\mu(1-\mu) \leq \frac{k}{d-k}\mu.$$

By Chebyshev's inequality,

$$\Pr[E_2] = \Pr[X_J = 0] \leq \Pr[|X_J - \mu| \geq \mu] \leq \frac{\text{Var}(X_J)}{\mu^2} \leq \frac{k}{(d-k)\mu}.$$

Substituting the parameters $k \leq d^{2\epsilon}$ and $\mu \geq d^{-\epsilon}$:

$$\Pr[E_2] \leq \frac{d^{2\epsilon}}{(d - d^{2\epsilon})d^{-\epsilon}} \approx \frac{d^{3\epsilon}}{d} = d^{3\epsilon-1}.$$

With $\epsilon = 1/5$, we have $\Pr[E_2] \lesssim d^{3/5-1} = d^{-2/5}$. For sufficiently large d, this probability is strictly less than $1/4$.

Conclusion: For sufficiently large d, $\Pr[E_1] + \Pr[E_2] < 1/4 + 1/4 = 1/2$. Consequently,

$$\Pr[R_J \cap (U \times V) \neq \emptyset] = 1 - \Pr[E_1 \cup E_2] \geq 1/2.$$

For small constant values of d where the asymptotic bounds do not yet hold, the constant ϵ can be chosen sufficiently small to satisfy the condition trivially or by adjusting the constants, as the existence of such an ϵ is guaranteed by the asymptotic behavior. □

I.5 Problem 8.5

I.5.1 Problem Statement

Theorem I.5. $\mathrm{Disj}(X_1, \ldots, X_n)$ *requires decision trees of depth* $n^2 - n$.

I.5.2 GPT

Proof. Let $n \geq 3$. For each ordered pair (i, j) with $1 \leq i, j \leq n$ and $i \neq j$, let $x_{i,j} \in \{0, 1\}$ be an input variable, and interpret $X_i = \{\, j \in [n] \setminus \{i\} : x_{i,j} = 1 \,\}$. Define

$$\mathrm{Disj}(X_1, \ldots, X_n) = 1 \quad \Longleftrightarrow \quad \forall j \in [n] \left| \{\, i \in [n] \setminus \{j\} : j \in X_i \,\} \right| \leq 1,$$

equivalently, $\mathrm{Disj}(X_1, \ldots, X_n) = 1$ if and only if for every $j \in [n]$ one has $\sum_{i \neq j} x_{i,j} \leq 1$. Let $m = n^2 - n$ be the number of variables.

We prove that every deterministic decision tree computing Disj has depth at least m. Since depth is always at most the number of variables, this will imply that the decision tree depth equals m.

Consider an arbitrary deterministic decision tree T that purports to compute Disj. We describe an adversary strategy for answering queries to the variables $x_{i,j}$ during the execution of T on an unknown input, and we show that under this strategy the computation cannot terminate at a leaf before querying all m variables.

For each column index $j \in [n]$, define the column

$$C_j = \{\, x_{i,j} : i \in [n] \setminus \{j\} \,\},$$

so $|C_j| = n - 1$. The adversary maintains a set $A \subseteq [n]$ of activated columns, initially $A = \varnothing$. Intuitively, a column j becomes activated once the adversary has answered 1 to some query in C_j. The adversary answers queries as follows.

Suppose the algorithm queries $x_{i,j}$. If $j \in A$, the adversary answers 0. If $j \notin A$, let u_j denote the number of as-yet-unqueried variables in C_j at the moment just before this query (so $u_j \geq 1$), and consider what u_j would become after answering 0 to this query, namely $u_j - 1$. If $u_j - 1 = 1$, the adversary answers 1 and sets $A := A \cup \{j\}$. Otherwise (i.e. $u_j - 1 \neq 1$), the adversary answers 0 and leaves A unchanged.

We first record an invariant implied by this strategy.

Claim. At every point in the interaction, for each $j \in [n]$ the following holds: if $j \notin A$, then either all variables in C_j have been queried, or at least two variables in C_j remain unqueried.

Proof of Claim. Fix j. While $j \notin A$, the adversary answers 0 to queries in C_j except possibly at the unique moment when answering 0 would leave exactly one unqueried variable in C_j; at that moment, instead of answering 0, the adversary answers 1 and adds j to A. Consequently, as long as $j \notin A$, the number of unqueried variables in C_j cannot equal 1. Since it is a nonnegative integer, it must be either 0 or at least 2. This proves the claim.

Now suppose, toward a contradiction, that the computation reaches a leaf after fewer than m queries. Let ρ be the (finite) root-to-leaf path taken, and let α be the partial assignment to queried variables determined by the adversary's answers along ρ. Since fewer than m variables were queried, there exists at least one unqueried variable. Choose any unqueried variable $x_{p,q} \in C_q$.

By construction of the strategy, for each activated column $j \in A$ there is exactly one queried variable in C_j that was answered 1, and all other queried variables in C_j were answered 0. For each nonactivated column $j \notin A$, every queried variable in C_j was answered 0.

We now build two total assignments $\beta^{(1)}$ and $\beta^{(0)}$ extending α such that Disj takes different values on them. This will contradict the assumption that the leaf reached by ρ is correctly labeled.

Construct $\beta^{(1)}$ as follows. For every unqueried variable $x_{i,j}$, set $x_{i,j} = 0$. Then, in each activated column $j \in A$, exactly one entry equals 1 (the one fixed to 1 by α) and all others equal 0, and in each nonactivated column all entries equal 0. Hence for every $j \in [n]$ we have $\sum_{i \neq j} x_{i,j} \leq 1$, so $\mathrm{Disj}(\beta^{(1)}) = 1$.

Next, construct $\beta^{(0)}$ by extending α as follows.

If $q \in A$, then set $x_{p,q} = 1$ and set every other unqueried variable to 0.

If $q \notin A$, then by the Claim the column C_q has at least two unqueried variables. Choose another unqueried variable $x_{r,q} \in C_q$ with $r \neq p$, and set $x_{p,q} = 1$, $x_{r,q} = 1$, and set every other unqueried variable to 0.

In either case, the assignment $\beta^{(0)}$ extends α because it agrees with α on all queried variables. Moreover, in the first case ($q \in A$), column q contains the original 1 fixed by α together with the newly set $x_{p,q} = 1$, so $\sum_{i \neq q} x_{i,q} \geq 2$; in the second case ($q \notin A$), column q contains two newly set 1's, so again $\sum_{i \neq q} x_{i,q} \geq 2$. Therefore $\mathrm{Disj}(\beta^{(0)}) = 0$.

Thus $\beta^{(1)}$ and $\beta^{(0)}$ are two inputs consistent with the same root-to-leaf transcript α but with different function values. Hence the leaf reached by ρ cannot be labeled to correctly output Disj on both inputs, contradicting correctness of T. This contradiction shows that no leaf can be reached under the adversary strategy before all m variables are queried, and therefore the depth of T is at least $m = n^2 - n$.

Since a decision tree can always query all m variables and then compute Disj, the minimum possible depth equals $m = n^2 - n$. □

I.5.3 Claude

Proof. We prove that the disjointness function $\mathrm{Disj}(X_1, \ldots, X_n)$, which determines whether the sets $X_1, \ldots, X_n \subseteq [m]$ are pairwise disjoint, requires decision trees of depth exactly $n^2 - n$.

First, we establish the lower bound. Consider any deterministic decision tree computing $\mathrm{Disj}(X_1, \ldots, X_n)$. The function outputs 1 if and only if for all pairs $i \neq j$, we have $X_i \cap X_j = \emptyset$.

To distinguish between a YES instance (all sets pairwise disjoint) and a NO instance (at least one pair intersects), the decision tree must verify the disjointness of all $\binom{n}{2} = \frac{n(n-1)}{2}$ pairs of sets.

For each pair (X_i, X_j) with $i \neq j$, there are m possible elements that could witness their intersection. In the worst case, to determine whether $X_i \cap X_j = \emptyset$, we need information about the membership of elements in both X_i and X_j.

Consider the adversary argument. Suppose the decision tree has made fewer than $n^2 - n$ queries. Then there exists at least one ordered pair (i, j) with $i \neq j$ such that for no element $e \in [m]$ have both memberships $e \in X_i$ and $e \in X_j$ been queried.

The adversary can maintain two consistent inputs: one where $X_i \cap X_j = \emptyset$ (a YES instance) and one where there exists e with $e \in X_i \cap X_j$ (a NO instance). Since the decision tree cannot distinguish these inputs with fewer than $n^2 - n$ queries, it cannot correctly compute Disj.

More precisely, for each ordered pair (i, j) with $i \neq j$, at least one query is needed to rule out the possibility that some element belongs to both X_i and X_j. There are $n(n-1) = n^2 - n$ such ordered pairs, yielding the lower bound.

For the upper bound, we construct a decision tree of depth $n^2 - n$. For each ordered pair (i, j) with $i \neq j$, we can query whether there exists any element in $X_i \cap X_j$ by systematically checking memberships. Specifically, for each pair, we select an arbitrary element $e_{i,j} \in [m]$ and query both $e_{i,j} \in X_i$ and $e_{i,j} \in X_j$. If both answers are yes for any pair, we output 0. If we exhaust all pairs without finding an intersection, we output 1.

This strategy uses at most $2 \cdot \binom{n}{2} = n(n-1) = n^2 - n$ queries in the worst case when we must verify all pairs.

Therefore, the decision tree complexity of $\mathrm{Disj}(X_1, \ldots, X_n)$ is exactly $n^2 - n$. □

I.5.4 Gemini

Proof. Let the input to the function $\text{Disj}(X_1, \ldots, X_n)$ be represented by a boolean matrix $M \in \{0,1\}^{n \times n}$, where $M_{ij} = 1$ if and only if the element j is contained in the set X_i. The universe of elements is assumed to be $\{1, \ldots, n\}$. The function is defined as:

$$\text{Disj}(M) = 1 \iff \bigcap_{i=1}^{n} X_i = \emptyset.$$

This condition is equivalent to stating that for every column $j \in \{1, \ldots, n\}$, there exists at least one row i such that $M_{ij} = 0$. Conversely, $\text{Disj}(M) = 0$ if and only if there exists a column j such that $M_{ij} = 1$ for all $i \in \{1, \ldots, n\}$.

We establish the lower bound on the decision tree depth using an adversary argument. Let T be any deterministic decision tree computing $\text{Disj}(M)$. We construct an adversary strategy that forces T to query every entry of the matrix M, thereby proving a lower bound of n^2, which implies the stated bound of $n^2 - n$.

Adversary Strategy: The adversary maintains a partial assignment of the matrix M. When the decision tree queries the value of a cell M_{ij}, the adversary responds according to the following rule:

1. If the cell M_{ij} is not the last unqueried cell in column j, the adversary answers 1.

2. If the cell M_{ij} is the last unqueried cell in column j (implying all other $n-1$ cells in column j have been queried and assigned 1), the adversary answers 0.

Correctness of the Lower Bound: We show that the decision tree T cannot terminate before querying all n^2 entries. Suppose, for the sake of contradiction, that T terminates after querying a set of cells Q with $|Q| < n^2$. Let S be the set of unqueried cells, so $S \neq \emptyset$. Let $(r, c) \in S$ be an unqueried cell in column c.

Based on the adversary strategy, a 0 is only revealed in a column when the algorithm queries the final bit of that column. Since (r, c) is unqueried, column c has not been fully queried. Therefore, all queries made so far to column c (if any) have resulted in the value 1.

The current information known to T is consistent with two valid inputs, $M^{(A)}$ and $M^{(B)}$, constructed as follows:

- **Input $M^{(A)}$:** Set all unqueried cells in S to 0. For every column j, if j was fully queried, the adversary strategy ensures it contains a 0. If j was not fully queried, the unqueried cells are set to 0, ensuring column j contains a 0. Thus, every column contains a 0, and $\text{Disj}(M^{(A)}) = 1$.

- **Input $M^{(B)}$:** Set all unqueried cells in column c to 1, and all other unqueried cells to 0. Since all previously queried cells in column c are 1 (by the adversary strategy) and we set the remaining unqueried cells in c to 1, column c consists entirely of 1s. Thus, $\bigcap_{i=1}^{n} X_i \neq \emptyset$, and $\text{Disj}(M^{(B)}) = 0$.

Since $M^{(A)}$ and $M^{(B)}$ agree on all queried positions Q, the deterministic decision tree T must produce the same output for both inputs. However, $\text{Disj}(M^{(A)}) \neq \text{Disj}(M^{(B)})$. This contradiction implies that T cannot terminate while any cell remains unqueried.

Consequently, any decision tree computing $\text{Disj}(X_1, \ldots, X_n)$ must have depth at least n^2. Since $n^2 \geq n^2 - n$ for all $n \geq 1$, the theorem holds. □

I.6 Problem 8.6

I.6.1 Problem Statement

Lemma I.6. *The function f can be certified using m^{2m+1} certificates of length $2m + m \log m$. Moreover, all the 1 inputs are certified unambiguously.*

I.6.2 GPT

Proof. Fix $m \in \mathbb{N}$. A certificate for a Boolean function is a partial assignment ρ to the input variables such that every total input x extending ρ has the same function value. We prove the lemma by exhibiting an explicit certificate family of the claimed size and length and then proving correctness and unambiguity on 1-inputs.

Assume that the input to f is organized so that there is a designated collection of $2m$ Boolean variables $(b_1, \ldots, b_{2m})$, and a designated collection of m address variables $(a_1, \ldots, a_m)$ where each a_j is represented in binary using exactly $\log m$ bits and hence ranges over $[m] = \{1, 2, \ldots, m\}$. Assume moreover that the definition of f is of the following form: there is a set of witnesses

$$W = [m]^{2m+1} = \{(t_0, t_1, \ldots, t_{2m}) : t_i \in [m]\},$$

and for each witness $t = (t_0, t_1, \ldots, t_{2m}) \in W$ there is a predicate P_t depending only on the variables $(b_1, \ldots, b_{2m}, a_1, \ldots, a_m)$ such that for every input x,

$$f(x) = 1 \iff \exists t \in W \text{ such that } P_t(x) = 1,$$

and furthermore this witnessing is unique in the sense that for every x with $f(x) = 1$ there is exactly one $t \in W$ such that $P_t(x) = 1$.

For each $t = (t_0, t_1, \ldots, t_{2m}) \in W$, define a partial assignment ρ_t as follows. First, for each $i \in \{1, \ldots, 2m\}$, set b_i to the unique Boolean value required by P_t for the variable b_i. Second, for each $j \in \{1, \ldots, m\}$, set the binary encoding of a_j to the unique value in $[m]$ required by P_t for the variable a_j. Since each a_j uses exactly $\log m$ bits, the assignment of all address variables fixes $m \log m$ input bits, and fixing $(b_1, \ldots, b_{2m})$ fixes an additional $2m$ input bits. Hence the length of ρ_t is exactly $2m + m \log m$.

Let $\mathcal{C} = \{\rho_t : t \in W\}$. Then $|\mathcal{C}| = |W| = m^{2m+1}$, establishing the stated upper bound on the number of certificates. It remains to show that each ρ_t is indeed a certificate and that the 1-inputs are certified unambiguously.

Fix $t \in W$. By construction, ρ_t assigns all variables on which P_t depends. Therefore, for any total input x extending ρ_t, the truth value of $P_t(x)$ is forced and equals 1, because ρ_t sets each involved variable to the value required for P_t to hold. Consequently, for any $x \succeq \rho_t$ we have $f(x) = 1$, since $P_t(x) = 1$ implies $f(x) = 1$ by the definition of f. This proves that each ρ_t is a valid 1-certificate for f.

Now let x be any input with $f(x) = 1$. By assumption, there exists a unique witness $t \in W$ such that $P_t(x) = 1$. Because ρ_t assigns precisely the variables occurring in P_t to the values required for P_t to be true, the condition $P_t(x) = 1$ implies that x extends ρ_t, i.e., $x \succeq \rho_t$. Thus every 1-input extends at least one certificate in $\mathcal{C}$.

To prove unambiguity on 1-inputs, suppose that a 1-input x extends two certificates ρ_t and $\rho_{t'}$ in $\mathcal{C}$. Then, by the certificate property established above, $P_t(x) = 1$ and $P_{t'}(x) = 1$, since extending ρ_t forces P_t to hold and extending $\rho_{t'}$ forces $P_{t'}$ to hold. By the uniqueness of the witness for x, this implies $t = t'$. Hence a 1-input extends at most one certificate in $\mathcal{C}$. Combined with the previous paragraph, every 1-input extends exactly one certificate in $\mathcal{C}$, which is precisely the claimed unambiguous certification of the 1-inputs.

Therefore, f can be certified using m^{2m+1} certificates of length $2m + m \log m$, and all 1-inputs are certified unambiguously. □

I.6.3 Claude

Proof. We construct a certificate scheme for the function f and analyze its properties.

Let the input domain be $\{0, 1\}^n$ where $n = 2m + m \log m$. We partition the inputs into two classes: those mapping to 1 and those mapping to 0 under f.

For each input x with $f(x) = 1$, we assign a unique certificate c_x of length $\ell = 2m + m \log m$. The certificate c_x encodes sufficient information to verify that x is indeed a 1-input. Specifically, the certificate contains:

- A witness of length $2m$ that provides direct evidence for $f(x) = 1$

- An index of length $m \log m$ that uniquely identifies which of the possible m^m configurations applies to this input

The total number of distinct certificates needed is bounded by m^{2m+1}. To see this, observe that we need at most one certificate per equivalence class of inputs that share the same verification structure. The number of such classes is determined by:

$$\text{Number of certificates} \leq m^{2m} \cdot m = m^{2m+1}$$

This bound arises because there are at most m^{2m} distinct witness patterns of length $2m$ over an alphabet of size m, and we require an additional factor of m to account for the indexing structure.

For unambiguous certification of 1-inputs, we ensure that each certificate c satisfies the following property: if a verifier receives input x and certificate c, and the verification accepts, then $f(x) = 1$ with certainty. This is achieved by constructing certificates such that:

$$\forall x, x' \in \{0,1\}^n : \text{ if } V(x,c) = 1 \text{ and } V(x',c) = 1, \text{ then } f(x) = f(x') = 1$$

where V denotes the verification algorithm. The certificate c is designed to be consistent only with inputs mapping to 1, ensuring no 0-input can be falsely certified.

The certificate length $\ell = 2m + m \log m$ is sufficient because:

- The $2m$ bits provide the witness information needed for verification
- The $m \log m$ bits provide enough entropy to distinguish among the m^m possible configurations, since $2^{m \log m} = m^m$

Therefore, we have constructed a certificate scheme using at most m^{2m+1} certificates, each of length $2m + m \log m$, with the property that all 1-inputs are certified unambiguously. □

I.6.4 Gemini

Proof. Let $\mathcal{C}$ denote the set of 1-certificates for the function f. We identify the certificates with the set of valid computation paths for f. Specifically, we associate each certificate $c \in \mathcal{C}$ with a sequence $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{2m+1}) \in [m]^{2m+1}$, where $[m] = \{1, \ldots, m\}$. This correspondence implies that the total number of distinct certificates is given by the cardinality of the set of sequences:

$$|\mathcal{C}| = \left|[m]^{2m+1}\right| = m^{2m+1}.$$

Next, we determine the length of each certificate. A certificate c_σ certifies a 1-input by fixing the values of specific input variables to ensure the computation follows the path σ. The structure of f dictates that verifying such a path requires querying two types of variables:

1. m pointer variables, each taking a value in $[m]$, which requires $\log m$ bits per variable.
2. $2m$ boolean variables, each taking a value in $\{0,1\}$, which requires 1 bit per variable.

Summing these components, the total bit-length L of a certificate is:

$$L = m \cdot \log m + 2m \cdot 1 = 2m + m \log m.$$

Finally, we establish the unambiguity of the certification. By the definition of the function f, for every input x such that $f(x) = 1$, the active pointers and boolean values in x define a unique valid path from the source to the sink. Consequently, there exists exactly one sequence $\sigma \in [m]^{2m+1}$ such that the certificate c_σ is consistent with x. Thus, every 1-input is certified by exactly one certificate from $\mathcal{C}$, proving that the certification is unambiguous. □

I.7 Problem 8.7

I.7.1 Problem Statement

Lemma I.7. *The decision tree complexity of f is $\Omega(m^2)$.*

I.7.2 GPT

Proof. Let $n = m^2$ and let $f : \{0,1\}^n \to \{0,1\}$ be the Boolean function defined by $f(x) = 1$ if and only if $x_i = 1$ for all $i \in \{1, \ldots, n\}$. Let T be any deterministic decision tree that computes f, and let d be the depth of T.

Consider the input $x^\star \in \{0,1\}^n$ defined by $x^\star = (1, 1, \ldots, 1)$. Since T computes f, the execution of T on input $x^\star$ follows a root-to-leaf path P and outputs 1 at the leaf reached, because $f(x^\star) = 1$.

For each index $j \in \{1, \ldots, n\}$, define an input $x^{(j)} \in \{0,1\}^n$ by $x_j^{(j)} = 0$ and $x_i^{(j)} = 1$ for all $i \neq j$. Then $f(x^{(j)}) = 0$.

We claim that along the path P, the decision tree T must query the variable x_j for every $j \in \{1, \ldots, n\}$. Suppose for contradiction that there exists some j such that x_j is not queried on P. Then for every query made along P, the queried index i satisfies $i \neq j$, and hence the answer returned by the oracle on input $x^{(j)}$ is the same as on input $x^\star$, namely 1, for each such queried index. Therefore the execution of T on input $x^{(j)}$ follows exactly the same path P and reaches the same leaf as on input $x^\star$. Consequently, T would output 1 on input $x^{(j)}$. This contradicts the fact that T computes f, since $f(x^{(j)}) = 0$.

Thus every variable x_j is queried along P. Hence the length of P is at least $n = m^2$, so the depth d of T satisfies $d \geq m^2$. Since T was arbitrary, the decision tree complexity of f satisfies $\mathrm{DT}(f) \geq m^2$, which is $\Omega(m^2)$. □

I.7.3 Claude

Proof. We prove that the decision tree complexity of f is $\Omega(m^2)$ by establishing a lower bound using an adversary argument.

Let $D(f)$ denote the decision tree complexity of f. We construct an adversary strategy that forces any deterministic decision tree to make at least $\Omega(m^2)$ queries in the worst case.

Consider the input space of f and partition it into equivalence classes based on the function's behavior. Without loss of generality, assume that f depends on n input variables and that there exist m critical variables whose values determine the output in a non-trivial manner.

We employ the following adversary strategy: The adversary maintains a set of consistent inputs that remain indistinguishable based on the queries made so far. Initially, this set contains all possible inputs. For each query made by the decision tree to variable x_i, the adversary responds in a way that maximizes the number of remaining consistent inputs.

Let S_t denote the set of consistent inputs after t queries. We claim that after $t < \frac{m^2}{4}$ queries, there exist at least two inputs $x, y \in S_t$ such that $f(x) \neq f(y)$.

To establish this claim, observe that the function f has a certificate complexity of at least m for some input. This means there exists an input x^* such that any certificate for $f(x^*)$ must specify at least m variables. Furthermore, the sensitivity of f at various inputs requires that changing certain subsets of variables affects the output.

Consider the $m \times m$ grid of variable pairs (i, j) where $1 \leq i, j \leq m$. Each query to the decision tree can eliminate at most one configuration from this grid in terms of the interaction between pairs of critical variables. Since there are $\binom{m}{2} = \frac{m(m-1)}{2} = \Omega(m^2)$ such pairs, and the function f exhibits dependence on the joint configuration of these pairs, any decision tree must query enough variables to distinguish between inputs that differ on these pairs.

More precisely, by a counting argument, if fewer than cm^2 queries are made for some sufficiently small constant $c > 0$, then there exist two inputs x and y in S_t such that $f(x) \neq f(y)$, but the decision tree has not yet queried enough variables to distinguish between them. This contradicts the correctness of the decision tree.

Therefore, any correct decision tree for f must make at least $\Omega(m^2)$ queries in the worst case, which establishes that $D(f) = \Omega(m^2)$. □

I.7.4 Gemini

Proof. We consider the case where f is the graph connectivity function defined on a graph $G = (V, E)$ with $|V| = m$. The input to f consists of $\binom{m}{2}$ binary variables representing the presence or absence of edges between vertices. We establish the lower bound using an adversary argument.

Let the adversary partition the vertex set V into two disjoint subsets V_1 and V_2 such that their sizes are as equal as possible, specifically $|V_1| = \lfloor m/2 \rfloor$ and $|V_2| = \lceil m/2 \rceil$. Let S be the set of all possible edges connecting a vertex in V_1 to a vertex in V_2. The cardinality of S is given by:

$$|S| = |V_1| \cdot |V_2| = \lfloor m/2 \rfloor \lceil m/2 \rceil \geq \frac{m^2 - 1}{4}.$$

The adversary employs the following strategy to answer the queries of the decision tree: 1. If the algorithm queries an edge $e = (u, v)$ such that $u, v \in V_1$ or $u, v \in V_2$, the adversary answers 1 (the edge exists). 2. If the algorithm queries an edge $e \in S$, the adversary answers 0 (the edge does not exist), unless e is the last unqueried edge in S.

Suppose the decision tree terminates before querying all edges in S. Let $U \subseteq S$ be the set of unqueried edges in S, with $U \neq \emptyset$. Based on the adversary's responses, all edges within V_1 and within V_2 exist, and all queried edges in S do not exist. Let $e^* \in U$ be an unqueried edge.

Consider two input graphs G_0 and G_1 consistent with the answers given so far and the unqueried variables: 1. In G_0, all edges in U are set to 0. The graph consists of two disjoint cliques on V_1 and V_2 with no edges between them. Thus, G_0 is disconnected and $f(G_0) = 0$. 2. In G_1, the edge e^* is set to 1 and all other edges in $U \setminus \{e^*\}$ are set to 0. The graph consists of two cliques connected by a single bridge e^*. Thus, G_1 is connected and $f(G_1) = 1$.

Since $f(G_0) \neq f(G_1)$, the decision tree cannot correctly determine the value of f without querying e^*. Because this holds for any unqueried edge in S, the algorithm must query every edge in S in the worst case. Therefore, the decision tree complexity $D(f)$ is bounded below by the size of S:

$$D(f) \geq |S| \geq \frac{m^2 - 1}{4}.$$

We conclude that the decision tree complexity of f is $\Omega(m^2)$. □

J Problems in Chapter 9

J.1 Problem 9.1

J.1.1 Problem Statement

Theorem J.1. *Every circuit of size s can be simulated by a formula of size 2^s.*

J.1.2 GPT

Proof. Fix a Boolean circuit C of size s. We assume the usual model in which C is a finite directed acyclic graph whose vertices are of the following types: input vertices labelled by variables $x_1, \ldots, x_n$ (and possibly the constants $0, 1$), and internal gate vertices labelled by Boolean connectives from a fixed finite basis (for concreteness, one may take $\{\wedge, \vee, \neg\}$), and with each gate having the appropriate fan-in for its label. The size of C is the number of internal gate vertices, which is s. The circuit has a designated output vertex. For a vertex v of C, denote by $\mathrm{val}_v(\alpha) \in \{0, 1\}$ the value computed at v on an input assignment $\alpha \in \{0, 1\}^n$. A Boolean formula is a rooted tree whose internal nodes are labelled by connectives from the basis and whose leaves are labelled by input variables or constants; its size is the number of internal nodes (equivalently, the number of connective occurrences).

We construct, for every vertex v of C, a formula Φ_v such that for every assignment α, the value of Φ_v on α equals $\mathrm{val}_v(\alpha)$, and such that the size of Φ_v is at most 2^s. In particular, for the output vertex o, Φ_o will be a formula computing the same Boolean function as C.

Choose a topological ordering of the vertices of C and define the depth of a vertex v to be the maximum number of internal gate vertices on any directed path starting at v and ending at the output vertex. Because the circuit is acyclic and has exactly s internal gate vertices, every directed path contains at most s internal gates, hence every vertex has depth at most s. We define formulas by backward induction on this depth.

If v is an input vertex labelled by x_i (or a constant $0, 1$), let Φ_v be the one-leaf formula consisting solely of that label, so $\Phi_v(\alpha) = \alpha_i$ (or the constant) for every α, and its size is 0.

If v is an internal gate vertex, let its label be a connective $\circ$ of arity $k \in \{1, 2\}$ (for the basis $\{\wedge, \vee, \neg\}$, $k = 1$ for $\neg$ and $k = 2$ for $\wedge, \vee$). Let its children (incoming neighbors) be $u_1, \ldots, u_k$. By induction, formulas $\Phi_{u_1}, \ldots, \Phi_{u_k}$ have been defined. Define Φ_v to be the formula whose root is labelled by $\circ$ and whose k immediate subformulas are $\Phi_{u_1}, \ldots, \Phi_{u_k}$ (for $k = 1$, there is one subformula). Then for every assignment α,

$$\Phi_v(\alpha) = \circ\big(\Phi_{u_1}(\alpha), \ldots, \Phi_{u_k}(\alpha)\big) = \circ\big(\mathrm{val}_{u_1}(\alpha), \ldots, \mathrm{val}_{u_k}(\alpha)\big) = \mathrm{val}_v(\alpha),$$

since the circuit semantics at v applies the connective $\circ$ to the values of its children.

It remains to bound the size of Φ_v. Let $|\Phi|$ denote the number of internal nodes of a formula Φ. Then by construction, if v is an internal gate of arity k, we have

$$|\Phi_v| = 1 + \sum_{j=1}^{k} |\Phi_{u_j}|.$$

For an input vertex v, $|\Phi_v| = 0$.

For an integer $d \geq 0$, define

$$M(d) = \max\{|\Phi_v| : v \text{ is a vertex of } C \text{ with depth at most } d\}.$$

We show by induction on d that $M(d) \leq 2^d - 1$ for all $d \geq 0$. For $d = 0$, any vertex of depth 0 cannot have an internal gate on a path to the output, hence it must be an input vertex, so $|\Phi_v| = 0$ and thus $M(0) = 0 \leq 2^0 - 1 = 0$.

Assume $d \geq 1$ and that $M(d-1) \leq 2^{d-1} - 1$. Let v be any vertex of depth at most d. If v is an input, then $|\Phi_v| = 0 \leq 2^d - 1$. If v is an internal gate of arity $k \leq 2$ with children $u_1, \ldots, u_k$, then each u_j has depth at most $d - 1$ (any path from u_j to the output extends to a path from v by adding the gate v, so the maximum number of internal gates on a path from u_j is at most one less). Hence $|\Phi_{u_j}| \leq M(d-1) \leq 2^{d-1} - 1$ for each j, and therefore

$$|\Phi_v| = 1 + \sum_{j=1}^{k} |\Phi_{u_j}| \leq 1 + k(2^{d-1} - 1) \leq 1 + 2(2^{d-1} - 1) = 2^d - 1.$$

Taking the maximum over all such v yields $M(d) \leq 2^d - 1$, completing the induction.

Let o be the output vertex of C. Its depth is at most s, since any directed path from o to the output is trivial and any path from any vertex to the output passes through at most s internal gates in total. In particular, the depth of o is at most s. Thus

$$|\Phi_o| \leq M(s) \leq 2^s - 1 \leq 2^s.$$

We have already shown that Φ_o computes the same Boolean function as C, so C is simulated by a formula of size at most 2^s, as claimed. □

J.1.3 Claude

Proof. Let C be a circuit of size s. We will construct a formula F that computes the same function as C and has size at most 2^s.

We proceed by induction on the structure of the circuit. The circuit C consists of s gates, where each gate computes a Boolean function of its inputs. We label the gates as $g_1, g_2, \ldots, g_s$ in topological order, so that if gate g_i depends on gate g_j, then $j < i$.

For each gate g_i in the circuit, we construct a formula F_i that computes the same function as the subcircuit rooted at g_i. We prove by induction on i that $|F_i| \leq 2^i$.

Base case: If g_i is an input gate, then F_i is simply the corresponding input variable, so $|F_i| = 1 \leq 2^i$.

Inductive step: Suppose g_i is not an input gate. Then g_i computes some Boolean operation (such as AND, OR, or NOT) applied to one or two predecessor gates. Let us consider the case where $g_i = g_j \wedge g_k$ for some $j, k < i$ (the cases for other operations are analogous).

By the inductive hypothesis, we have formulas F_j and F_k computing the functions of gates g_j and g_k respectively, with $|F_j| \leq 2^j$ and $|F_k| \leq 2^k$.

We construct $F_i = F_j \wedge F_k$ by taking the conjunction of F_j and F_k. The size of this formula is:

$$|F_i| = |F_j| + |F_k| + 1 \leq 2^j + 2^k + 1.$$

Since $j, k < i$ and $i \leq s$, we have $j, k \leq i - 1$. Therefore:

$$|F_i| \leq 2^{i-1} + 2^{i-1} + 1 = 2 \cdot 2^{i-1} + 1 = 2^i + 1.$$

However, this bound is not tight enough. To obtain the correct bound, we observe that in the worst case, when we construct the formula for the output gate g_s, we may need to duplicate subformulas. The key observation is that each gate in the circuit may have fan-out greater than 1, meaning its output may be used by multiple other gates.

To handle arbitrary fan-out, whenever a gate g_i is used by multiple gates, we substitute the entire formula F_i into each location where g_i is referenced. This substitution can cause exponential blow-up.

More precisely, let d_i denote the number of times gate g_i is used in the computation leading to the output gate. Then the size of the formula for gate g_i after all substitutions is bounded by $\prod_{j<i} d_j^{c_j}$ where c_j indicates the contribution of gate g_j to gate g_i.

In the worst case, the circuit forms a complete binary tree of depth $\log_2 s$, where each gate has fan-out 2. In this case, the formula size grows as $2^{\text{depth}} = 2^{\log_2 s} = s$ for a tree, but for a general DAG with reconvergent fan-out, the blow-up can be exponential.

The precise bound is obtained by observing that each of the s gates can be duplicated in the formula construction, and in the worst case (a chain of gates each with fan-out 2), the formula size doubles at each level. Since there are at most s levels, the final formula size is at most 2^s.

Therefore, the formula $F = F_s$ corresponding to the output gate has size at most 2^s and computes the same function as the circuit C. □

J.1.4 Gemini

Proof. We proceed by induction on the size of the circuit, denoted by s. We assume the standard basis of gates $\{\wedge, \vee, \neg\}$, where $\wedge$ and $\vee$ have fan-in 2 and $\neg$ has fan-in 1. We define the size of a circuit as the number of gates (nodes) in the directed acyclic graph, and the size of a formula as the number of leaves (variable occurrences) in its tree representation.

Base Case: Let $s = 1$. The circuit consists of a single node, which must be an input variable or a constant. The equivalent formula is the variable or constant itself. The size of this formula is 1. Since $1 \leq 2^1$, the base case holds.

Inductive Step: Assume that for any circuit of size $k < s$, there exists an equivalent formula of size at most 2^k. Let C be a circuit of size s with output gate g. We consider the structure of g:

Case 1: $g = \neg u$. Let C_u be the sub-circuit rooted at u. The size of C_u, denoted s_u, satisfies $s_u \leq s - 1$ because g is not contained in C_u. By the inductive hypothesis, there exists a formula F_u equivalent to C_u with size $|F_u| \leq 2^{s_u}$. We construct the formula $F = \neg F_u$. The size of F is equal to the size of F_u (as negation does not increase the number of leaves). Thus:

$$|F| = |F_u| \leq 2^{s_u} \leq 2^{s-1} < 2^s.$$

Case 2: $g = u * v$, where $* \in \{\wedge, \vee\}$. Let C_u and C_v be the sub-circuits rooted at u and v, respectively. Let s_u and s_v denote their sizes. Since g is the parent of u and v, g is not contained in either sub-circuit. Therefore, $s_u \leq s - 1$ and $s_v \leq s - 1$. Note that C_u and C_v may share nodes, but the formula construction unrolls the circuit into a tree, duplicating shared sub-structures.

By the inductive hypothesis, there exist formulas F_u and F_v equivalent to C_u and C_v respectively, such that $|F_u| \leq 2^{s_u}$ and $|F_v| \leq 2^{s_v}$. We construct the formula $F = F_u * F_v$. The size of F is the sum of the sizes of its sub-formulas:

$$|F| = |F_u| + |F_v|.$$

Substituting the bounds from the inductive hypothesis:

$$|F| \leq 2^{s_u} + 2^{s_v} \leq 2^{s-1} + 2^{s-1} = 2 \cdot 2^{s-1} = 2^s.$$

In all cases, we have constructed an equivalent formula of size at most 2^s. □

J.2 Problem 9.2

J.2.1 Problem Statement

Theorem J.2. *There exists a Boolean function $f : \{0,1\}^n \to \{0,1\}$ that can be computed by a circuit of depth $O(\sqrt{n})$, but cannot be computed by any circuit of depth $O(n^{1/4})$.*

J.2.2 GPT

Proof. Fix any absolute constants $c_1, c_2 > 0$ with $c_2 > c_1$, and for each $n \in \mathbb{N}$ define

$$d_1(n) = \lceil c_1 n^{1/4} \rceil, \qquad d_2(n) = \lceil c_2 \sqrt{n} \rceil.$$

Throughout, by a circuit on n inputs we mean a (nonuniform) family of Boolean circuits over the standard basis $\{\wedge, \vee, \neg\}$ of polynomial size in n (equivalently, size at most n^k for some fixed k independent of n), with unbounded fan-in $\wedge, \vee$ and fan-in 1 for $\neg$, and depth measured in the usual way as the maximum number of gates on any input-to-output path.

We use the following depth hierarchy theorem for polynomial-size Boolean circuits.

Lemma. There exists an absolute constant $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ and all integers $1 \leq d < D$,

there exists a Boolean function $g : \{0,1\}^n \to \{0,1\}$ computable by a polynomial-size circuit of depth at most D,

but not computable by any polynomial-size circuit of depth at most d.

Assuming the lemma, we complete the proof of the theorem as follows.

Choose $n \geq n_0$ large enough so that $1 \leq d_1(n) < d_2(n)$, which holds for all sufficiently large n since $n^{1/4} = o(\sqrt{n})$. Apply the lemma with $d = d_1(n)$ and $D = d_2(n)$. Then there exists a Boolean function $f : \{0,1\}^n \to \{0,1\}$ such that:

(i) f is computable by a polynomial-size circuit of depth at most $d_2(n) = \lceil c_2 \sqrt{n} \rceil$,

(ii) f is not computable by any polynomial-size circuit of depth at most $d_1(n) = \lceil c_1 n^{1/4} \rceil$.

Property (i) implies that f can be computed by a circuit of depth $O(\sqrt{n})$, and property (ii) implies that f cannot be computed by any circuit of depth $O(n^{1/4})$ (within the standard convention of polynomial size). This establishes the stated separation for the chosen n, and hence proves the existence claim.

Therefore, there exists a Boolean function $f : \{0,1\}^n \to \{0,1\}$ that can be computed by a circuit of depth $O(\sqrt{n})$ but cannot be computed by any circuit of depth $O(n^{1/4})$. □

J.2.3 Claude

Proof. We construct an explicit Boolean function that witnesses the separation between depth $O(\sqrt{n})$ and depth $O(n^{1/4})$.

Let $m = \lfloor \sqrt{n} \rfloor$ and consider the input $x \in \{0,1\}^n$ partitioned into m blocks of size approximately $n/m \approx \sqrt{n}$ each. Specifically, we define the function f as follows:

For $i = 1, 2, \ldots, m$, let B_i denote the i-th block of variables, where $B_i = \{x_{(i-1)\cdot\lceil n/m \rceil + 1}, \ldots, x_{\min(i\cdot\lceil n/m \rceil, n)}\}$. Define

$$f(x) = \bigvee_{i=1}^{m} \left(\bigwedge_{j \in B_i} x_j \right).$$

That is, $f(x) = 1$ if and only if there exists at least one block in which all variables are set to 1.

First, we show that f can be computed by a circuit of depth $O(\sqrt{n})$. The computation proceeds in two stages:

- Stage 1: For each block B_i, compute $\bigwedge_{j \in B_i} x_j$ using an AND gate with fan-in $|B_i| \leq \lceil n/m \rceil \leq \lceil \sqrt{n} \rceil + 1 = O(\sqrt{n})$. This requires depth 1.
- Stage 2: Compute the OR of all m outputs from Stage 1 using an OR gate with fan-in $m = \lfloor \sqrt{n} \rfloor = O(\sqrt{n})$. This requires depth 1.

Thus, the total depth is $2 = O(1)$ when using unbounded fan-in gates, or $O(\log \sqrt{n}) = O(\log n)$ with bounded fan-in. However, for the standard model with fan-in 2, we can compute each block AND in depth $O(\log \sqrt{n}) = O(\log n)$ and the final OR in depth $O(\log m) = O(\log \sqrt{n}) = O(\log n)$, giving total depth $O(\log n)$.

More precisely, using the standard depth measure with unbounded fan-in, the depth is exactly 2. With bounded fan-in, we achieve depth $O(\sqrt{n})$ by using a different construction: we compute the function using a circuit where each AND gate has fan-in 2 and we build a balanced tree of depth $O(\log \sqrt{n}) = O(\log n) \ll O(\sqrt{n})$.

Now we prove the lower bound. Suppose for contradiction that f can be computed by a circuit C of depth $d = c \cdot n^{1/4}$ for some sufficiently small constant $c > 0$.

Consider the communication complexity of f under the following partition: Alice receives the first $m/2$ blocks and Bob receives the last $m/2$ blocks. By the structure of f, Alice and Bob must determine whether Alice has an all-ones block or Bob has an all-ones block.

Using the depth-to-communication complexity relationship, a circuit of depth d can be simulated by a communication protocol with communication at most $O(n/2^d)$ bits. For $d = c \cdot n^{1/4}$, this gives communication $O(n/2^{c \cdot n^{1/4}})$.

However, the communication complexity of f under this partition is $\Omega(m) = \Omega(\sqrt{n})$. This is because in the worst case, Alice must communicate which of her $m/2$ blocks (if any) is all-ones, requiring $\Omega(\log m) = \Omega(\log \sqrt{n})$ bits, or Bob must verify all of Alice's blocks, requiring $\Omega(m \cdot \sqrt{n}) = \Omega(n)$ bits in total information transfer.

More rigorously, by a counting argument on the number of possible functions computable at depth d, we have that the number of distinct functions on n variables computable by depth-d circuits is at most $2^{O(n \cdot 2^d)}$. For our function f to be computable at depth $d = c \cdot n^{1/4}$, we need $2^{O(n \cdot 2^{c \cdot n^{1/4}})} \geq 2^{2^n}$, which requires $n \cdot 2^{c \cdot n^{1/4}} \geq 2^n$. For sufficiently large n and small c, this inequality fails, yielding a contradiction.

Therefore, f cannot be computed by any circuit of depth $O(n^{1/4})$, completing the proof. □

J.2.4 Gemini

Proof. We prove the existence of such a function using a counting argument on a restricted number of variables. Let n be a sufficiently large integer. We define a parameter $m = \lfloor \sqrt{n} \rfloor$. We consider Boolean functions on m variables, which can be viewed as functions on n variables that depend only on the first m coordinates.

First, we establish a lower bound on the circuit depth required to compute almost all Boolean functions on m variables. Let $\mathcal{B}_m$ denote the set of all Boolean functions $g : \{0,1\}^m \to \{0,1\}$. The size of this set is $|\mathcal{B}_m| = 2^{2^m}$.

Consider Boolean circuits with fan-in 2 over the standard basis $\{\wedge, \vee, \neg\}$. A circuit of depth d can be expanded into a formula (a tree) of depth d. The number of leaves in a binary tree of depth d is at most 2^d. Each leaf is labeled with a literal from $\{x_1, \ldots, x_m, \neg x_1, \ldots, \neg x_m\}$ or a constant $\{0,1\}$. There are $2m + 2$ choices for each leaf. Each internal node is labeled with a gate from $\{\wedge, \vee\}$. There are fewer than 2^d internal nodes.

Let $N(m,d)$ be the number of distinct functions on m variables computable by a circuit of depth d. We can bound $N(m,d)$ by the number of distinct labeled trees of depth d. A loose upper bound is sufficient for our purposes:

$$N(m,d) \leq (2m+2)^{2^d} \cdot 2^{2^d} = (4m+4)^{2^d}.$$

We seek a depth d such that $N(m,d) < |\mathcal{B}_m|$. Taking logarithms base 2:

$$\log_2 N(m,d) \leq 2^d \log_2(4m+4).$$

We require $2^d \log_2(4m+4) < 2^m$. This inequality holds if:

$$d + \log_2 \log_2(4m+4) < m.$$

Let $d_{max} = m - \lceil \log_2 \log_2(4m+4) \rceil - 1$. For this depth, the number of computable functions is strictly less than the total number of functions. Therefore, there exists a function $g \in \mathcal{B}_m$ that cannot be computed by any circuit of depth d_{max}. Since $m = \lfloor \sqrt{n} \rfloor$, for sufficiently large n, we have:

$$\text{depth}(g) \geq m - O(\log \log m) = \Omega(\sqrt{n}).$$

Now, we define the function $f : \{0,1\}^n \to \{0,1\}$ by $f(x_1, \ldots, x_n) = g(x_1, \ldots, x_m)$. The value of f does not depend on $x_{m+1}, \ldots, x_n$.

Upper Bound: Any Boolean function on m variables can be computed by a Disjunctive Normal Form (DNF) formula. A DNF formula for g consists of an OR of at most 2^m terms, where each term is an AND of m literals. Using a tree of OR gates, the fan-in 2^m can be handled in depth $\lceil \log_2(2^m) \rceil = m$. Using a tree of AND gates, the fan-in m can be handled in depth $\lceil \log_2 m \rceil$. Thus, g (and consequently f) can be computed by a circuit of depth:

$$D(f) \leq m + \lceil \log_2 m \rceil = O(m) = O(\sqrt{n}).$$

Lower Bound: Suppose for the sake of contradiction that f can be computed by a circuit family of depth $O(n^{1/4})$. That is, there exists a constant $c > 0$ such that for sufficiently large n, $\text{depth}(f) \leq cn^{1/4}$. Let C_n be a circuit computing f with depth bounded by $cn^{1/4}$. By setting the input variables $x_{m+1}, \ldots, x_n$ to the constant 0, we obtain a circuit C'_n that computes g on inputs $x_1, \ldots, x_m$. The depth of C'_n is no greater than the depth of C_n. Therefore, g is computable in depth $cn^{1/4}$. However, we established that g requires depth $\Omega(\sqrt{n})$. Specifically, for large n:

$$\text{depth}(g) \geq \lfloor \sqrt{n} \rfloor - O(\log \log \sqrt{n}).$$

Since $\lim_{n\to\infty} \frac{cn^{1/4}}{\sqrt{n}} = 0$, for sufficiently large n, $cn^{1/4} < \lfloor \sqrt{n} \rfloor - O(\log \log \sqrt{n})$. This contradicts the assumption that f is computable in depth $O(n^{1/4})$.

Thus, there exists a Boolean function f computable in depth $O(\sqrt{n})$ but not in depth $O(n^{1/4})$. □

J.3 Problem 9.3

J.3.1 Problem Statement

Theorem J.3. *Any Boolean formula can be* balanced*. If a Boolean function can be computed by a formula of size s, then it can be computed using a formula of depth $O(\log s)$.*

J.3.2 GPT

Proof. Let f be a Boolean function that can be computed by a formula of size s. We aim to show that f can also be computed by a formula of depth $O(\log s)$.

First, recall that a Boolean formula is composed of AND, OR, and NOT gates, where each gate has a fixed arity. The size s of a formula refers to the total number of gates in the formula.

Consider a binary tree representation of the formula. Each gate corresponds to a node in the tree, with its children representing the inputs to the gate. The size of the formula is the number of nodes in this tree, and the depth of the formula corresponds to the length of the longest path from the root to a leaf.

The key observation is that any Boolean formula can be balanced, which means we can construct a binary tree where all leaves are at approximately the same depth. In other words, we can reorganize the gates of the formula in such a way that the depth of the tree is minimized, while maintaining the same size.

To achieve this, we use the following approach. Consider a binary tree of size s. A complete binary tree with s nodes has a depth of approximately $\log_2 s$. This can be seen by noting that in a complete binary tree, the number of nodes at depth d is 2^d, and the total number of nodes is the sum of the nodes at all depths. For a tree with s nodes, the depth must satisfy the relation $2^d \approx s$, which implies $d = O(\log s)$.

By constructing a binary tree of this form, we can ensure that the Boolean formula has depth $O(\log s)$ while preserving its size s. Therefore, any Boolean formula of size s can be computed by a formula of depth $O(\log s)$.

Thus, we have shown that any Boolean formula can be balanced, and a Boolean function computable by a formula of size s can also be computed by a formula of depth $O(\log s)$. □

J.3.3 Claude

Proof. We prove that any Boolean formula of size s can be transformed into an equivalent formula of depth $O(\log s)$.

Let F be a Boolean formula of size s, where size is measured as the number of leaves (variables or constants) in the formula tree. We proceed by induction on the structure of the formula.

First, we establish a key lemma about balancing binary trees. Consider the formula tree T corresponding to F. Each internal node represents a Boolean operation (AND, OR, or NOT), and each leaf represents a variable or constant.

For any subtree T' of T with k leaves, we show that T' can be restructured to have depth $O(\log k)$ while preserving the computed function.

Base case: If $k = 1$, the subtree is a single leaf with depth 0, and $O(\log 1) = O(0) = 0$.

Inductive step: Assume the claim holds for all subtrees with fewer than k leaves, where $k \geq 2$. Consider a subtree T' with k leaves and root operation $\circ \in \{\wedge, \vee\}$ (we handle NOT separately).

The root of T' has two subtrees T_1 and T_2 with k_1 and k_2 leaves respectively, where $k_1 + k_2 = k$. By the inductive hypothesis, we can balance T_1 to depth $d_1 \leq c \log k_1$ and T_2 to depth $d_2 \leq c \log k_2$ for some constant $c > 0$.

If $|d_1 - d_2| \leq 1$, the combined depth is $\max(d_1, d_2) + 1 \leq c \log k + 1 \leq (c + 1) \log k$ for sufficiently large k.

If $|d_1 - d_2| > 1$, without loss of generality assume $d_1 > d_2 + 1$. We restructure using the associativity of $\circ$. Let $T_1 = T_{1L} \circ T_{1R}$ where T_{1L} and T_{1R} are the left and right subtrees of T_1's root. We can rewrite:

$$(T_{1L} \circ T_{1R}) \circ T_2 = T_{1L} \circ (T_{1R} \circ T_2)$$

By repeatedly applying this restructuring, we can ensure that the depths of the two subtrees at each node differ by at most 1. This process maintains the invariant that each subtree with m leaves has depth at most $\lceil \log_2 m \rceil + O(1)$.

For NOT gates, since they are unary operations, they add at most 1 to the depth and do not affect the balancing argument asymptotically.

Now we apply this construction to the entire formula F with s leaves. The balanced formula has depth:

$$d \leq \lceil \log_2 s \rceil + c$$

for some constant c accounting for the restructuring overhead.

Therefore, $d = O(\log s)$.

The transformation preserves the computed Boolean function because we only use associativity and commutativity of AND and OR operations, which are valid equivalences in Boolean algebra.

Thus, any Boolean function computable by a formula of size s can be computed by a formula of depth $O(\log s)$. □

J.3.4 Gemini

Proof. Let ϕ be a Boolean formula of size s, where the size $S(\phi) = s$ is defined as the number of leaves (literals) in the binary tree representing the formula. We define the depth $D(\phi)$ as the length of the longest path from the root to a leaf. We aim to show that there exists an equivalent formula ϕ' such that $D(\phi') = O(\log s)$. We proceed by induction on s.

For the base case, if $s = 1$, the formula consists of a single literal. The depth is 0 (or 1 depending on convention), which satisfies $O(1)$.

For $s > 1$, we first establish a lemma regarding the existence of a separator in binary trees. Let T be the binary tree corresponding to ϕ. We claim there exists a subtree T_v rooted at a node v such that its size

$s_v = S(T_v)$ satisfies $s/3 \leq s_v \leq 2s/3$. To prove this, start at the root of T. While the current node has a child with a subtree size strictly greater than $2s/3$, traverse to that child. This path must terminate at a node u such that $S(u) > 2s/3$, but for both children u_L and u_R, $S(u_L) \leq 2s/3$ and $S(u_R) \leq 2s/3$. Since $S(u) = S(u_L) + S(u_R)$, it is impossible for both children to have size less than $S(u)/2$. Thus, at least one child has size at least $S(u)/2 > (2s/3)/2 = s/3$. Let v be this child. Then $s/3 < S(v) \leq 2s/3$.

Let ψ be the subformula of ϕ corresponding to the subtree T_v. We construct a new formula using the Shannon expansion (or multiplexer logic) based on the output of ψ. Let $\phi|_{\psi=1}$ denote the formula ϕ where the subformula ψ is replaced by the constant 1, and $\phi|_{\psi=0}$ denote ϕ where ψ is replaced by the constant 0. The original function computed by ϕ is equivalent to:

$$\Phi \equiv (\psi \wedge \phi|_{\psi=1}) \vee (\neg\psi \wedge \phi|_{\psi=0})$$

Let $D(s)$ be the maximum depth required for a formula of size s. The depth of Φ is determined by the maximum depth of its components plus the constant depth required for the connecting $\wedge$, $\vee$, and $\neg$ gates (which is at most 2). Thus:

$$D(s) \leq \max(D(S(\psi)), D(S(\phi|_{\psi=1})), D(S(\phi|_{\psi=0}))) + 2$$

We analyze the sizes of the components: 1. $S(\psi) \leq 2s/3$. 2. The size of $\phi|_{\psi=1}$ is at most $s - S(\psi) + 1$ (the $+1$ accounts for the constant leaf replacing the subtree). Since $S(\psi) \geq s/3$, we have $S(\phi|_{\psi=1}) \leq s - s/3 + 1 = 2s/3 + 1$. 3. Similarly, $S(\phi|_{\psi=0}) \leq 2s/3 + 1$.

Therefore, the recurrence relation for the depth is:

$$D(s) \leq D(\lfloor 2s/3 \rfloor + 1) + 2$$

For sufficiently large s, $\lfloor 2s/3 \rfloor + 1 \leq cs$ for any constant $c \in (2/3, 1)$. The depth of the recursion tree is $O(\log_{3/2} s)$, and at each level of the recursion, we add a constant amount of depth. Solving this recurrence yields:

$$D(s) = O(\log s)$$

Thus, any Boolean function computable by a formula of size s can be computed by a formula of depth $O(\log s)$. □

J.4 Problem 9.4

J.4.1 Problem Statement

Theorem J.4. *We showed that any formula that computes whether or not $x \in [2n]^n$ corresponds to n distinct numbers requires a formula of size $\Omega(n^2)$. This gives a Boolean function that depends on m bits but requires $\Omega(m^2/\log^2 m)$ size formulas. Show how you can improve the lower bound to get a Boolean function depending on m bits that requires formulas of size $\Omega(m^2/\log m)$.*

Hint: Consider the element distinctness function with $x \in [n]^k$ for the appropriate n and k.

J.4.2 GPT

Proof. We begin by considering the element distinctness function, where the input is a set $x \in [n]^k$ of k elements, and we want to determine whether all the elements in the set are distinct. The element distinctness function can be viewed as a Boolean function $f : \{0,1\}^m \to \{0,1\}$, where $m = k \log n$, representing the k elements as binary strings of length $\log n$. Our goal is to show that this function requires a formula of size $\Omega(m^2/\log m)$.

First, we note that the element distinctness function is known to have a lower bound of $\Omega(k^2)$ for a set of size k. Specifically, it is well-known that any formula that computes whether a set of k elements from $[n]$ are distinct requires a formula of size $\Omega(k^2/\log k)$. We now relate this lower bound to the size of the Boolean formula required to compute the element distinctness function for the set $x \in [n]^k$.

Let $m = k \log n$ be the total number of bits required to represent the k elements in the set. The goal is to express the lower bound for the size of the formula in terms of m. Observe that if we had a formula of

size smaller than $\Omega(k^2/\log k)$, it would contradict the known lower bound for the distinctness function. Hence, we must establish a relationship between k and m.

Since $m = k \log n$, we have $k = m/\log n$. Substituting this into the known lower bound for the distinctness function, we get:

$$\Omega\left(\left(\frac{m}{\log n}\right)^2\right) = \Omega\left(\frac{m^2}{\log^2 n}\right)$$

Next, we optimize this bound. The key observation is that n can be chosen as a function of m, specifically $n = 2^m$. This choice allows us to reduce the logarithmic factor in the denominator. If we set $n = 2^m$, then $\log n = m$. Substituting this into the bound, we obtain:

$$\Omega\left(\frac{m^2}{\log^2 m}\right)$$

Thus, we have improved the lower bound from $\Omega(m^2/\log^2 m)$ to the desired lower bound of $\Omega(m^2/\log m)$.

Therefore, we conclude that the Boolean function corresponding to the element distinctness problem, with m bits of input, requires a formula of size $\Omega(m^2/\log m)$. □

J.4.3 Claude

Proof. We will improve the lower bound from $\Omega(m^2/\log^2 m)$ to $\Omega(m^2/\log m)$ by carefully choosing the parameters for the element distinctness function.

Let k and n be parameters to be determined. Consider the element distinctness function $\text{ED}_{n,k}$ on inputs $x \in [n]^k$, which outputs 1 if and only if all k elements of x are distinct.

The input size is $m = k \log n$ bits, since we need $\log n$ bits to represent each of the k elements from $[n]$.

From the known lower bound for element distinctness, any formula computing $\text{ED}_{n,k}$ requires size $\Omega(k^2)$ when $k \leq n$. This is because the formula must essentially compare all pairs of elements, requiring $\Omega(k^2)$ operations.

Now we choose parameters to optimize the lower bound in terms of m. Set $k = n$, so we consider $x \in [n]^n$.

With this choice:

- The input size is $m = n \log n$ bits
- The formula size lower bound is $\Omega(n^2)$

From $m = n \log n$, we can solve for n in terms of m. We have $n = \Theta(m/\log n)$. By substitution, $n = \Theta(m/\log(m/\log n))$. Since $\log(m/\log n) = \log m - \log\log n = \Theta(\log m)$ when $n = \Theta(m/\log m)$, we obtain $n = \Theta(m/\log m)$.

Therefore, the formula size lower bound becomes:

$$\Omega(n^2) = \Omega\left(\left(\frac{m}{\log m}\right)^2\right) = \Omega\left(\frac{m^2}{\log^2 m}\right)$$

To improve this to $\Omega(m^2/\log m)$, we instead choose $k = n/\log n$.

With this choice:

- The input size is $m = k \log n = \frac{n}{\log n} \cdot \log n = n$ bits
- The formula size lower bound is $\Omega(k^2) = \Omega(n^2/\log^2 n)$

From $m = n$, we have $n = m$, so:

$$\Omega(k^2) = \Omega\left(\frac{n^2}{\log^2 n}\right) = \Omega\left(\frac{m^2}{\log^2 m}\right)$$

This still gives the original bound. The correct approach is to set $k = n$ but use a more refined analysis. Actually, we reconsider: set $k = \sqrt{n \log n}$ and choose n appropriately.

The key insight is to balance the parameters differently. Set $n = 2^{\sqrt{m}}$ and $k = \sqrt{m}/\log n = \sqrt{m}/\sqrt{m} = 1$, which is too small.

The correct choice is: Let $k = m/\log m$ and $n = 2^{\log m} = m$.

Then the input size is $k \log n = \frac{m}{\log m} \cdot \log m = m$ bits, and the formula size is $\Omega(k^2) = \Omega(m^2/\log^2 m)$.

To achieve $\Omega(m^2/\log m)$, we use the fact that for $k = \Theta(\sqrt{m \log m})$ and $n = 2^{\sqrt{m/\log m}}$, we have $m = k \log n = \Theta(\sqrt{m \log m} \cdot \sqrt{m/\log m}) = \Theta(m)$, and the lower bound is $\Omega(k^2) = \Omega(m \log m)$, which is still insufficient.

The resolution is that the improved bound $\Omega(m^2/\log m)$ follows from a more careful analysis of the element distinctness function using communication complexity or more refined formula lower bound techniques, which show that the quadratic dependence on k can be achieved with only a single logarithmic factor loss when parameters are optimally chosen. □

J.4.4 Gemini

Proof. Let $ED_{n,k} : [n]^k \to \{0, 1\}$ be the element distinctness function which accepts a sequence of k integers $x_1, \ldots, x_k$ where each $x_i \in \{1, \ldots, n\}$, and outputs 1 if and only if all x_i are pairwise distinct. We encode each input x_i using $b = \lceil \log_2 n \rceil$ bits. The total number of input bits for the Boolean function is $m = k \cdot b \approx k \log n$.

We utilize Nechiporuk's Theorem to establish a lower bound on the formula size $L(ED_{n,k})$. Nechiporuk's Theorem states that if the set of input variables X is partitioned into disjoint sets $R_1, \ldots, R_k$, then

$$L(f) \geq \frac{1}{4} \sum_{i=1}^{k} \log_2 N_i,$$

where N_i is the number of distinct subfunctions on the variables in R_i obtained by fixing the variables in $X \setminus R_i$ to constants.

We partition the m bits into k blocks $R_1, \ldots, R_k$, where R_i contains the b bits representing the integer x_i. To estimate N_i, we consider the number of distinct subfunctions on x_i generated by fixing the other $k - 1$ variables x_j (for $j \neq i$) to specific values.

Let $S = \{v_1, \ldots, v_{k-1}\}$ be a subset of $\{1, \ldots, n\}$ of size $k - 1$. We can fix the variables x_j for $j \neq i$ such that their values correspond to the distinct elements of S. Under this assignment, the function becomes a function of x_i alone, denoted $g_S(x_i)$, which evaluates to 1 if and only if $x_i \notin S$. Since the zero-set of g_S is exactly S, distinct sets S yield distinct subfunctions. The number of such subsets is $\binom{n}{k-1}$. Therefore, $N_i \geq \binom{n}{k-1}$.

Applying the theorem, we obtain:

$$L(ED_{n,k}) \geq \frac{1}{4} \sum_{i=1}^{k} \log_2 \binom{n}{k-1} = \frac{k}{4} \log_2 \binom{n}{k-1}.$$

Using the bound $\log_2 \binom{n}{r} \geq r \log_2(n/r)$, and assuming $n \geq k$, we have:

$$L(ED_{n,k}) = \Omega\left(k \cdot k \log\left(\frac{n}{k}\right)\right) = \Omega\left(k^2 \log\left(\frac{n}{k}\right)\right).$$

To improve the lower bound relative to m, we choose $n = k^2$. With this choice, the number of bits is:

$$m = k\lceil \log_2(k^2) \rceil \approx 2k \log_2 k.$$

This implies $\log m \approx \log k + \log\log k = \Theta(\log k)$. Substituting $n = k^2$ into the lower bound expression:

$$L(ED_{k^2,k}) = \Omega\left(k^2 \log\left(\frac{k^2}{k}\right)\right) = \Omega(k^2 \log k).$$

Now we express this bound in terms of m. We calculate the target complexity class $\Omega(m^2/\log m)$:

$$\frac{m^2}{\log m} \approx \frac{(2k\log k)^2}{\log k} = \frac{4k^2\log^2 k}{\log k} = 4k^2 \log k = \Theta(k^2 \log k).$$

Since $L(ED_{k^2,k}) = \Omega(k^2 \log k)$, it follows that:

$$L(ED_{k^2,k}) = \Omega\left(\frac{m^2}{\log m}\right).$$

Thus, by choosing the domain size $n = k^2$, the element distinctness function provides the desired lower bound. □

J.5 Problem 9.5

J.5.1 Problem Statement

Theorem J.5. *Suppose $\phi(x_1, \ldots, x_n, y_1, \ldots, y_n)$ is a Boolean formula in conjunctive normal form that is unsatisfiable. Suppose Alice and Bob are given $x, y \in \{0,1\}^n$, respectively, and want to find a specific clause in ϕ that is not satisfied.*

1. Show that if the formula has a resolution refutation of depth d, then Alice and Bob can find this clause with d bits of communication using a deterministic protocol.

2. Show that if the formula has a cutting-planes refutation of depth d, then Alice and Bob can find this clause with $d \log d$ bits of communication using a randomized protocol.

J.5.2 GPT

Proof. Let $\phi(x_1, \ldots, x_n, y_1, \ldots, y_n)$ be an unsatisfiable CNF. Write $\phi = \bigwedge_{i=1}^m C_i$, where each clause C_i is a disjunction of literals over the variables $x_1, \ldots, x_n, y_1, \ldots, y_n$. For inputs $x \in \{0,1\}^n$ (held by Alice) and $y \in \{0,1\}^n$ (held by Bob), define the search relation

$$R_\phi = \{(x, y, i) \in \{0,1\}^n \times \{0,1\}^n \times [m] : C_i(x, y) = 0\},$$

where $C_i(x, y) = 0$ means the clause C_i is falsified by the complete assignment (x, y).

1. Assume ϕ has a resolution refutation of depth d. Fix a resolution derivation tree T of depth d whose leaves are clauses from ϕ and whose root is the empty clause $\perp$. Each internal node is labeled by a resolvent obtained from its two children by resolving on some variable $v \in \{x_1, \ldots, x_n, y_1, \ldots, y_n\}$. Thus if an internal node is labeled by clause D, it has children labeled by $D \vee v$ and $D \vee \neg v$ for some variable v, and D is their resolvent. The depth of the refutation is the maximum number of resolution steps along any root-to-leaf path, hence the height of T is at most d.

For any total assignment $\alpha \in \{0,1\}^{2n}$ to all variables, define a deterministic walk from the root to a leaf as follows. At an internal node labeled by D with children labeled by $D \vee v$ and $D \vee \neg v$, let $\alpha(v) \in \{0,1\}$ be the value of v under α. If $\alpha(v) = 1$, proceed to the child labeled by $D \vee \neg v$; if $\alpha(v) = 0$, proceed to the child labeled by $D \vee v$. This choice is well-defined and depends only on $\alpha(v)$. Along the walk, after choosing the corresponding child, the following invariant holds: if the current node is labeled by clause E, then $E(\alpha) = 0$. This is proved by induction on the distance from the root. At the root the label is $\perp$, which is falsified by every assignment. Suppose the invariant holds at a node labeled D, so $D(\alpha) = 0$, and we resolve on v to children $D \vee v$ and $D \vee \neg v$. If $\alpha(v) = 0$, then $(D \vee v)(\alpha) = D(\alpha) \vee \alpha(v) = 0 \vee 0 = 0$, so moving to $D \vee v$ preserves falsity. If $\alpha(v) = 1$, then $(D \vee \neg v)(\alpha) = D(\alpha) \vee (1 - \alpha(v)) = 0 \vee 0 = 0$, so moving to $D \vee \neg v$ preserves falsity. Therefore the label at the reached leaf is a clause from ϕ that is falsified by α, hence it is an index i with $C_i(\alpha) = 0$.

Now set α to be the concatenation of Alice's and Bob's inputs, i.e., $\alpha(x_j) = x_j$ and $\alpha(y_j) = y_j$. It remains to implement the above walk by a deterministic communication protocol and bound its communication by d bits. At each internal node the walk requires the value of the pivot variable v. If $v = x_j$, Alice knows $\alpha(v) = x_j$ and can send one bit equal to x_j to Bob; both parties then determine the next child. If $v = y_j$, Bob analogously sends one bit y_j to Alice. If desired, one may instead fix that the party who knows the pivot bit transmits it; the other party updates the current node label accordingly. Each step uses exactly one communicated bit, and the walk length is at most d, so the total communication is at most d bits. The protocol outputs the leaf clause index i reached by this walk, and by the invariant this clause is not satisfied by (x, y). This proves the first claim.

2. Assume ϕ has a cutting-planes refutation of depth d. We use the standard translation of clauses into linear inequalities over variables in $\{0, 1\}$. For each Boolean variable z (each x_j and y_j), we interpret $z \in \{0, 1\}$ and include the Boolean axioms $0 \le z \le 1$. A clause $C = \ell_1 \vee \cdots \vee \ell_k$ is encoded as the inequality $\sum_{t=1}^{k} s(\ell_t) \ge 1$, where $s(z) = z$ and $s(\neg z) = 1 - z$. Under any Boolean assignment, the inequality is satisfied if and only if the clause is satisfied.

A cutting-planes proof derives from the initial inequalities (the encoded clauses and Boolean axioms) new inequalities by applying either nonnegative linear combination, or division with rounding: from $\sum_i a_i u_i \ge b$ with integer coefficients a_i and integer $t \ge 1$, infer $\sum_i \lceil a_i/t \rceil u_i \ge \lceil b/t \rceil$ after suitable normalization; equivalently, from an inequality with integer coefficients, infer its division by a positive integer followed by rounding up. A refutation ends by deriving an explicit contradiction, e.g., $0 \ge 1$. The depth d means there is a derivation DAG whose longest path from an initial inequality to the contradiction has length at most d, and such that each inference step increases depth by 1.

Fix a depth-d cutting-planes refutation DAG G whose sink is the contradictory inequality $0 \ge 1$. We describe a public-coin randomized protocol with communication $O(d \log d)$ that on input (x, y) outputs a falsified clause of ϕ. The protocol maintains a current derived inequality I appearing as a node of G such that $I(x, y)$ is false. Initially take I to be the final contradiction $0 \ge 1$, which is false under every assignment. The protocol iteratively moves backward along one inference step to a predecessor inequality that remains false under (x, y), until it reaches an initial inequality. If the initial inequality is a Boolean axiom, then that axiom must be falsified, which is impossible for Boolean inputs; hence the reached initial inequality must be an encoded clause inequality, and thus identifies a clause C_i falsified by (x, y).

It suffices to implement one backward move with communication $O(\log d)$ in such a way that with probability 1 a predecessor inequality that is false under (x, y) is selected, and that the number of moves is at most d. Consider an inference producing an inequality I from predecessors $I_1, \ldots, I_r$ by a rule of cutting planes. We treat two cases.

Case A: I is obtained as a nonnegative integer linear combination $I = \sum_{j=1}^{r} \lambda_j I_j$ with $\lambda_j \in \mathbb{Z}_{\ge 0}$. Write each I_j as $A_j \cdot u \ge b_j$, where u lists all variables and A_j is an integer coefficient vector. Then I is $A \cdot u \ge b$ where $A = \sum_j \lambda_j A_j$ and $b = \sum_j \lambda_j b_j$. Under a fixed assignment (x, y), define the slack value $\sigma_j = (A_j \cdot u)(x, y) - b_j$ and $\sigma = (A \cdot u)(x, y) - b$. Then $\sigma = \sum_j \lambda_j \sigma_j$. If I is false under (x, y), then $\sigma < 0$. Since each $\lambda_j \ge 0$, it follows that there exists some j with $\sigma_j < 0$, i.e., I_j is also false under (x, y). Therefore, to move backward it suffices to find an index j such that $I_j(x, y)$ is false.

To find such a j with small communication, use public randomness to pick an integer modulus p uniformly at random from a set of primes $\mathcal{P}$ of size M, where M will be chosen below. Both parties can compute, from their own input halves, the value of each left-hand side modulo p by splitting the dot product into x-part and y-part. More explicitly, for each inequality I_j with left-hand side $A_j \cdot u$, write $A_j \cdot u = A_{j,x} \cdot x + A_{j,y} \cdot y$, where $A_{j,x}$ and $A_{j,y}$ are the restrictions of coefficients to the x- and y-variables, with constant terms absorbed by extending u with a 1 if convenient. Alice computes $v_{j,x} = (A_{j,x} \cdot x) \bmod p$ and Bob computes $v_{j,y} = (A_{j,y} \cdot y) \bmod p$. With one message of $\lceil \log p \rceil$ bits from Alice to Bob (or vice versa), they can combine to obtain $(A_j \cdot u)(x, y) \bmod p$ for any requested j. However we need to identify some falsified predecessor among at most r candidates with only $O(\log d)$ communication overall per step. We use the fact that the depth is d, hence along any path there are at most d backward steps, and we may assume without loss of generality that each node in G has in-degree at most d by expanding fan-in with dummy combination steps; thus $r \le d$ at each inference.

Define the integer deficits $\delta_j = b_j - (A_j \cdot u)(x, y) = -\sigma_j$ and $\delta = b - (A \cdot u)(x, y) = -\sigma$. Then

I false means $\delta \geq 1$ and $\delta = \sum_j \lambda_j \delta_j$. We need some j with $\delta_j \geq 1$. Let $S = \{j : \delta_j \geq 1\}$. Note $S \neq \emptyset$. Consider the sum $T = \sum_{j=1}^{r} \lambda_j \delta_j$; then $T = \delta \geq 1$. Over the integers modulo p, we have $T \bmod p \equiv \sum_j \lambda_j (b_j - (A_j \cdot u)(x, y)) \bmod p$. Using public randomness, choose p such that $p > 2dB$ where B bounds the absolute values of all coefficients and right-hand sides appearing at depth at most d in the proof; such a bound exists and depends only on the fixed proof. Then $|\delta_j| \leq dB$ and $|T| \leq d \cdot dB = d^2 B$, and by choosing $p > 2d^2 B$ we ensure that congruence modulo p determines the sign of any such integer from its residue in $[0, p-1]$ as being 0 or at least 1 when interpreted appropriately. Since the proof is fixed, we can precompute a set $\mathcal{P}$ of primes all exceeding $2d^2 B$ and of size $M = d^2$; by Bertrand's postulate there are sufficiently many primes in $[2d^2 B, 4d^2 B]$ for large enough d, and for small d we may hardcode. For such p, the residue of δ_j equals δ_j itself in $\{0, 1, \ldots, p-1\}$ whenever $\delta_j \geq 0$ and $|\delta_j| < p$. Hence $\delta_j \geq 1$ iff $\delta_j \bmod p \in \{1, \ldots, p-1\}$.

Now the parties can identify a falsified predecessor by the following randomized selection. Using public randomness, pick a uniformly random $j \in \{1, \ldots, r\}$ and also pick a random prime $p \in \mathcal{P}$. Alice sends $(A_{j,x} \cdot x) \bmod p$ to Bob using $O(\log p) = O(\log(dB)) = O(\log d)$ bits (since B is fixed for the proof, $\log(dB) = O(\log d)$). Bob computes $(A_j \cdot u)(x, y) \bmod p$, hence $\delta_j \bmod p = (b_j - (A_j \cdot u)(x, y)) \bmod p$, and therefore can determine whether $\delta_j \geq 1$ (equivalently, whether I_j is false). If I_j is false, they move to predecessor I_j. If I_j is true, they repeat with fresh public randomness. Since $S \neq \emptyset$, the probability that a uniformly random j lands in S is $|S|/r \geq 1/r \geq 1/d$. Therefore the expected number of trials is at most d, and each trial communicates $O(\log d)$ bits; thus the expected communication per backward move is $O(d \log d)$, which would be too large. To obtain $O(\log d)$ communication per backward move, we instead perform a single trial but amplify success across the depth by allowing bounded error overall; equivalently, we accept a small failure probability and use repetition logarithmic in d at each move.

Specifically, at each backward move perform $t = \lceil c \log d \rceil$ independent trials, each trial choosing a fresh random j and prime p, and communicating the residue as above. If any trial finds a falsified predecessor I_j, move to such an I_j (choose the first found). If none succeed, output a special failure symbol. The failure probability of one trial is at most $1 - 1/d$, so the failure probability of t trials is at most $(1 - 1/d)^t \leq e^{-t/d} \leq e^{-c(\log d)/d}$. This bound is not strong enough. Instead, we use a distribution on j biased by the coefficients λ_j to guarantee constant success probability.

Define weights $w_j = \lambda_j$ and let $W = \sum_{j=1}^{r} w_j$. Sample j with probability w_j/W using public randomness. Then

$$\Pr[j \in S] = \sum_{j \in S} \frac{w_j}{W} = \frac{\sum_{j \in S} \lambda_j}{\sum_{j=1}^{r} \lambda_j}.$$

Since $\delta = \sum_j \lambda_j \delta_j \geq 1$ and each $\delta_j \leq dB$ in absolute value, we have

$$1 \leq \sum_{j \in S} \lambda_j \delta_j \leq (dB) \sum_{j \in S} \lambda_j,$$

so $\sum_{j \in S} \lambda_j \geq 1/(dB)$. Also $W = \sum_j \lambda_j$ is at most some fixed bound $W_{\max}$ depending on the proof and depth, and in particular one may assume each combination step is normalized so that $W \leq d$ by expanding steps if necessary. Hence $\Pr[j \in S] \geq 1/(d^2 B)$, which is still not constant, but now the required number of repetitions to make failure probability at most $1/(3d)$ per move is $t = O(\log(d)/(\Pr[j \in S])) = O(d^2 \log d)$, again too large.

Therefore we take a different approach standard in communication complexity of cutting planes: use the Karchmer–Wigderson game for threshold functions and the fact that a depth-d cutting-planes proof yields a depth-$O(d)$ threshold decision tree separating satisfying from falsifying assignments, which in turn yields a randomized protocol of cost $O(d \log d)$. We now formalize this reduction.

For each inequality I in the proof, define the Boolean predicate $f_I : \{0,1\}^{2n} \to \{0,1\}$ by $f_I(\alpha) = 1$ iff I is satisfied by α. The final contradiction has $f_{0 \geq 1} \equiv 0$. Each inference step of cutting planes corresponds to a rule that makes f_I a monotone threshold consequence of the predecessor predicates. In particular, the division rule introduces a threshold gate whose fan-in is the number of variables appearing, and the combination rule corresponds to summing weighted inputs. It is a standard fact that a depth-d

cutting-planes refutation yields a depth-d Boolean circuit of threshold gates computing the constant 0 function from the clause predicates and the Boolean axioms, and moreover, by restricting to Boolean inputs, the circuit can be arranged so that each internal gate computes a linear threshold function of its inputs with integer weights bounded by $\mathrm{poly}(d)$ and fan-in at most $\mathrm{poly}(d)$. Fix such a depth-d threshold circuit C whose inputs are the indicators of satisfaction of the initial inequalities (equivalently, of the initial clauses and axioms), and whose output is constantly 0 on all Boolean assignments. Since the circuit is a refutation, for every Boolean assignment α there is at least one input clause indicator equal to 0, i.e., some clause is falsified.

Consider the natural evaluation of C on the assignment $\alpha = (x, y)$. Since the output is 0, there exists a path in the circuit from the output to an input wire such that at each gate along the path the chosen child wire evaluates in a way that certifies the parent gate evaluates to 0. For a threshold gate g of fan-in k, with inputs $z_1, \ldots, z_k \in \{0, 1\}$ and integer weights $a_1, \ldots, a_k$ and threshold θ, the gate outputs 1 iff $\sum_{i=1}^{k} a_i z_i \geq \theta$. If $g(\alpha) = 0$, then $\sum_{i=1}^{k} a_i z_i < \theta$. A certificate for this strict inequality is given by an index i such that changing z_i from its current value to 1 would increase the sum but still not reach θ, or more generally by exhibiting a set of indices where $z_i = 0$ that accounts for the deficit; however to obtain a low-communication protocol we use a randomized subprotocol for the Karchmer–Wigderson game of threshold functions.

Define the Karchmer–Wigderson relation for C as follows: Alice holds $\alpha \in f^{-1}(1)$ and Bob holds $\beta \in f^{-1}(0)$ and must find an input bit where they differ. Here f is the function computed by the circuit. In our setting $f \equiv 0$, so we instead use the search version: Alice holds (x, y) split, and they must find an initial clause falsified. For a depth-d threshold circuit, there is a public-coin randomized protocol that, given a gate whose output on α is 0, finds a child input wire that is 0 with communication $O(\log k)$ where k is the fan-in, by sampling according to a distribution proportional to the absolute weights and using modular hashing to verify contributions; repeating over depth yields $O(d \log k)$, and with $k \leq \mathrm{poly}(d)$ gives $O(d \log d)$.

We now make this precise for one threshold gate g of fan-in $k \leq d^{c_0}$ and weights a_i with $|a_i| \leq d^{c_0}$ and threshold θ with $|\theta| \leq d^{c_0}$, for some constant c_0 depending only on the fixed refutation. Let the input bits be $z_i = f_{I_i}(\alpha)$ for predecessor inequalities I_i. Suppose $g(\alpha) = 0$, so $\sum_i a_i z_i < \theta$. Let $S_1 = \{i : z_i = 1\}$ and $S_0 = \{i : z_i = 0\}$. Define the deficit $\Delta = \theta - \sum_i a_i z_i \geq 1$. Let $A^+ = \sum_{i \in S_0,\, a_i > 0} a_i$ and $A^- = \sum_{i \in S_1,\, a_i < 0} (-a_i)$. Then

$$\theta - \sum_i a_i z_i = \theta - \sum_{i \in S_1} a_i = \theta - \sum_{i : a_i > 0} a_i z_i - \sum_{i : a_i < 0} a_i z_i.$$

Rewriting yields

$$\Delta = \theta - \sum_{i : a_i > 0} a_i z_i + \sum_{i : a_i < 0} (-a_i) z_i.$$

Since $z_i \in \{0, 1\}$, we have $\sum_{i : a_i > 0} a_i z_i \leq \sum_{i : a_i > 0} a_i$ and $\sum_{i : a_i < 0} (-a_i) z_i \leq \sum_{i : a_i < 0} (-a_i)$. Moreover, the deficit Δ can be expressed as

$$\Delta = \theta - \sum_{i : a_i > 0} a_i + \sum_{i \in S_0,\, a_i > 0} a_i + \sum_{i \in S_1,\, a_i < 0} (-a_i),$$

so

$$\sum_{i \in S_0,\, a_i > 0} a_i + \sum_{i \in S_1,\, a_i < 0} (-a_i) = \Delta - \theta + \sum_{i : a_i > 0} a_i.$$

The right-hand side is a positive integer bounded by $\mathrm{poly}(d)$. Hence at least one of the two sums is positive, implying either there exists i with $a_i > 0$ and $z_i = 0$, or there exists i with $a_i < 0$ and $z_i = 1$. In the first case $I_i(\alpha)$ is false (since z_i indicates satisfaction), and in the second case $I_i(\alpha)$ is true, but flipping it would decrease the sum. For our downward search we need a child with output 0, i.e., some i with $z_i = 0$. The existence of such i is immediate if all $a_i \geq 0$, which can be ensured by standard normalization of cutting-planes to monotone threshold gates on satisfaction bits. We assume this monotone form, so

$a_i \geq 0$ and g outputs 1 iff $\sum_i a_i z_i \geq \theta$ with $a_i, \theta \geq 0$. Then $g(\alpha) = 0$ implies $\sum_i a_i z_i \leq \theta - 1$, hence $\sum_{i \in S_0} a_i \geq \sum_i a_i - (\theta - 1)$, and in particular $S_0 \neq \emptyset$.

To find an index $i \in S_0$ with $O(\log k)$ communication, use the public-coin protocol that samples i with probability proportional to a_i and checks whether $z_i = 0$. Since $\sum_{i \in S_0} a_i \geq 1$ and $\sum_i a_i \leq k \cdot d^{c_0} \leq d^{c_0 + c_0}$, the success probability is at least $1/d^{O(1)}$. Repeating $O(\log d)$ times yields success probability at least $2/3$ for that gate. Each check of whether $z_i = 0$ amounts to checking whether inequality I_i is satisfied by α, which can be done with $O(\log d)$ communication by evaluating the linear form modulo a random prime of size $\mathrm{poly}(d)$ as above (Alice sends her partial sum modulo p to Bob, who compares with the threshold). Therefore each repetition costs $O(\log d)$ bits, and with $O(\log d)$ repetitions we get $O((\log d)^2)$ per gate. Since the fan-in $k \leq \mathrm{poly}(d)$, we can compress by choosing p of size $O(d^{O(1)})$ so that $\log p = O(\log d)$, and selecting i itself can be done without communication using public randomness.

Traversing from the output gate down to an input over depth at most d multiplies the cost by d. Thus the total communication is at most $O(d(\log d)^2)$. Finally, observe that by standard balancing one can ensure $\log k = O(\log d)$ and absorb one $\log d$ factor to obtain $O(d \log d)$ total communication; concretely, replace the per-gate repetition count by $\lceil c \log d \rceil$ and note that the check of z_i can be performed with $O(1)$ rounds and $O(\log d)$ bits, while the failure probability over d gates is at most $1/3$ by a union bound if each gate fails with probability at most $1/(3d)$, which is achieved with $O(\log d)$ repetitions when success probability is at least $1/d^{O(1)}$ by adjusting constants in the normalization of weights to ensure success probability at least $1/2$ via weight-splitting into unit weights, which increases fan-in to at most $\sum_i a_i \leq \mathrm{poly}(d)$ and makes the sampling uniform over at most $\mathrm{poly}(d)$ children, yielding success probability at least $1/\mathrm{poly}(d)$ and thus $O(\log d)$ repetitions.

Therefore there exists a public-coin randomized protocol that, given (x, y), follows the depth-d threshold circuit corresponding to the cutting-planes refutation downward to an input corresponding to an initial clause inequality that is falsified by (x, y), using at most $O(d \log d)$ bits of communication and erring with probability at most $1/3$. This establishes the second claim. □

J.5.3 Claude

Proof. We prove both parts of the theorem separately.

Part 1: Resolution refutation with d bits of communication.

Let $\phi(x_1, \ldots, x_n, y_1, \ldots, y_n)$ be an unsatisfiable CNF formula with a resolution refutation of depth d. A resolution refutation is a tree where leaves are clauses of ϕ and internal nodes are obtained by the resolution rule. The root is the empty clause, and the depth is the maximum distance from root to any leaf.

We construct a deterministic protocol as follows. Alice and Bob maintain a current clause C in the refutation tree, starting at the root (empty clause). They traverse the tree from root to leaf using the following procedure:

At each internal node with clause C obtained by resolving clauses C_1 and C_2 on variable z, exactly one of the following holds:

- If $z \in \{x_1, \ldots, x_n\}$, Alice knows the value of z in her assignment. She sends 1 bit indicating which child clause C_1 or C_2 to follow (the one that would be falsified by her partial assignment).
- If $z \in \{y_1, \ldots, y_n\}$, Bob knows the value of z in his assignment. He sends 1 bit indicating which child clause to follow.

Since the assignment (x, y) falsifies ϕ, at each resolution step, at least one of the parent clauses must be falsified by the current assignment. The party who controls the resolution variable determines which branch to follow by checking which clause is falsified.

This process continues for at most d steps (the depth of the refutation tree), with exactly 1 bit communicated per step. At the end, they reach a leaf clause of ϕ that is falsified by (x, y).

Therefore, the total communication is at most d bits.

Part 2: Cutting-planes refutation with $d \log d$ bits of communication.

Let ϕ have a cutting-planes refutation of depth d. A cutting-planes refutation derives linear inequalities from the clauses of ϕ using:

- Initial inequalities from clauses
- Linear combinations with non-negative coefficients
- Division and rounding

The refutation derives a contradiction (e.g., $0 \geq 1$).

We use a randomized protocol. The key observation is that we can use the polynomial method. Each clause C in ϕ corresponds to a linear inequality. For a falsified clause, the corresponding inequality is violated by (x, y).

The protocol works as follows:

Alice and Bob use shared randomness to choose a random prime p of size $O(\log d)$. They simulate the cutting-planes refutation modulo p, maintaining a current inequality at each step.

At each step i (for $i = 1, \ldots, d$):

- They maintain an inequality $\sum_j a_j z_j \geq b$ where z_j are variables
- Alice computes her contribution: $\sum_{j:z_j \in \{x_1,\ldots,x_n\}} a_j x_j \pmod p$
- Bob computes his contribution: $\sum_{j:z_j \in \{y_1,\ldots,y_n\}} a_j y_j \pmod p$
- They exchange $O(\log p) = O(\log d)$ bits to determine which inequality in the derivation is violated

By the Schwartz-Zippel lemma, if we choose p large enough (polynomial in d), the protocol succeeds with high probability in identifying a violated inequality at each step.

After d steps, they trace back to find an original clause of ϕ that is falsified. The total communication is $d \cdot O(\log d) = O(d \log d)$ bits.

By choosing the prime appropriately and using standard derandomization techniques if needed, the protocol uses $d \log d$ bits of communication. □

J.5.4 Gemini

Proof. Let $\phi(x, y)$ be an unsatisfiable Boolean formula in conjunctive normal form (CNF) over variables $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$. Let Alice hold an assignment $x \in \{0,1\}^n$ and Bob hold $y \in \{0,1\}^n$. Since ϕ is unsatisfiable, for any pair (x, y), there exists at least one clause $C \in \phi$ such that $C(x, y) = 0$. The problem of finding such a clause is the search problem associated with ϕ, denoted $Search(\phi)$.

Part 1: Resolution Refutation

Assume there exists a resolution refutation of ϕ of depth d. A resolution refutation is a directed acyclic graph (DAG) where:

1. The leaves (sources) are clauses of ϕ.
2. The root (sink) is the empty clause $\perp$.
3. Every internal node C is derived from two parents C_1 and C_2 via the resolution rule: $C = (C_1 \setminus \{z\}) \vee (C_2 \setminus \{\neg z\})$, where z is a literal corresponding to a variable in x or y.

We describe a deterministic protocol for Alice and Bob to find a falsified clause. The protocol maintains a "current clause" C_{curr} in the refutation DAG. The invariant maintained is that C_{curr} evaluates to False (0) under the joint assignment (x, y).

Protocol:

1. **Initialization:** Set $C_{curr} = \perp$. The empty clause is logically false, so the invariant holds at the start.
2. **Step:** While C_{curr} is not a leaf of the DAG:
   - Let C_{curr} be derived from parents C_1 and C_2 by resolving on variable v. Without loss of generality, assume C_1 contains v and C_2 contains $\neg v$.

- Since C_{curr} is False, all literals in C_{curr} are False.
- The players must determine which parent is False. Note that $C_1 = C_{curr} \vee v$ and $C_2 = C_{curr} \vee \neg v$.
- If v is a variable in x (say x_i), Alice sends the value of x_i to Bob. If v is in y (say y_j), Bob sends the value of y_j to Alice.
- If the value of v is 0 (False), then $C_1 = 0 \vee 0 = 0$. The players update $C_{curr} \leftarrow C_1$.
- If the value of v is 1 (True), then $\neg v = 0$, so $C_2 = 0 \vee 0 = 0$. The players update $C_{curr} \leftarrow C_2$.

3. **Termination:** When C_{curr} is a leaf, it corresponds to a clause in ϕ. By the invariant, this clause is not satisfied by (x, y). The players output this clause.

Complexity Analysis: The depth of the refutation is d. In each step, the players traverse from a node to one of its parents, moving one level up the tree (or along the edges of the DAG away from the sink). Thus, there are at most d steps. In each step, exactly 1 bit is communicated (the value of the resolution variable). The total communication is d bits.

Part 2: Cutting-Planes Refutation

Assume there exists a cutting-planes (CP) refutation of ϕ of depth d. In this system, clauses are translated into linear inequalities of the form $\sum a_i x_i + \sum b_j y_j \geq k$. A clause $\bigvee_{i \in I} z_i \vee \bigvee_{j \in J} \neg z_j$ becomes $\sum_{i \in I} z_i + \sum_{j \in J} (1 - z_j) \geq 1$. The refutation ends with a contradiction $0 \geq 1$.

The derivation rules are:

- *Addition:* From I_1 and I_2, derive $I_1 + I_2$.
- *Division:* From $\sum c_k z_k \geq k$, derive $\sum c_k z_k \geq \lceil k/c \rceil$ provided c divides all c_k.

We describe a randomized protocol. The invariant is that the current inequality I_{curr} is falsified by (x, y). *Protocol:*

1. **Initialization:** Set I_{curr} to the root contradiction $0 \geq 1$, which is clearly falsified.

2. **Step:** While I_{curr} is not a leaf (an initial clause):

   - If I_{curr} is derived by Division from I_{prev}, then if I_{curr} is False, I_{prev} must also be False (since division tightens the constraint, the logical implication is $I_{prev} \implies I_{curr}$, so $\neg I_{curr} \implies \neg I_{prev}$). Update $I_{curr} \leftarrow I_{prev}$. No communication is required.
   - If I_{curr} is derived by Addition from I_1 and I_2, let I_{curr} be $L_1(x, y) + L_2(x, y) \geq k_1 + k_2$, where $I_1 : L_1(x, y) \geq k_1$ and $I_2 : L_2(x, y) \geq k_2$.
   - Since I_{curr} is falsified, $L_1(x, y) + L_2(x, y) < k_1 + k_2$. This implies that either $L_1(x, y) < k_1$ or $L_2(x, y) < k_2$.
   - The players need to check if I_1 is falsified. I_1 has the form $\sum a_i x_i + \sum b_j y_j \geq k_1$.
   - Alice computes $v_A = \sum a_i x_i$. Bob computes $v_B = \sum b_j y_j$. They must check if $v_A + v_B \geq k_1$, or equivalently $v_A \geq k_1 - v_B$.
   - This is an instance of the Greater-Than problem (GT).
   - If I_1 is falsified, update $I_{curr} \leftarrow I_1$. Else, update $I_{curr} \leftarrow I_2$ (which must be falsified).

3. **Termination:** Output the leaf clause corresponding to I_{curr}.

Complexity Analysis: The depth of the proof is d. The number of Addition steps is at most d. We must bound the size of the coefficients to determine the cost of the GT protocol. In a CP proof of depth d, the coefficients and the constant term can grow by at most a factor of 2 at each addition step. Starting with small constants (from the clauses), the maximum value of any coefficient or partial sum at depth d is bounded by $2^{O(d)}$. Consequently, the values v_A and $k_1 - v_B$ can be represented using $m = O(d)$ bits.

The randomized communication complexity of the Greater-Than problem for two m-bit integers is $O(\log m)$. Here, $m = O(d)$, so the cost per addition step is $O(\log d)$. Since there are at most d steps requiring communication, the total communication complexity is:

$$d \times O(\log d) = O(d \log d).$$

Thus, Alice and Bob can find the unsatisfied clause with $O(d \log d)$ bits of communication. □

J.6 Problem 9.6

J.6.1 Problem Statement

Theorem J.6. *In this exercise we give another lower bound for the cutting planes proof system. Consider the game where Alice is given a permutation $\pi : [n] \to [n]$, and Bob is given a subset $Y \subseteq [n]$, with the promise that $\pi(1) \in Y, \pi(n) \notin Y$. Their goal is to compute i such that $\pi(i) \in Y, \pi(i+1) \notin Y$. One can show that $\Omega(\log^2 n)$ communication is required.*

1. Give a protocol that solves the game with $O(\log^2 n)$ communication.

2. Consider the monotone function that takes a graph G as input, and outputs whether or not the vertex 1 is connected to 2. Use the lower bound described above to prove that any monotone formula for this function must be of size at least $n^{\Omega(\log n)}$.

J.6.2 GPT

Proof. For $k \in \{1, \ldots, n\}$ define $b(k) \in \{0, 1\}$ by $b(k) = 1$ if and only if $\pi(k) \in Y$. The promise implies $b(1) = 1$ and $b(n) = 0$. Hence there exists $i \in \{1, \ldots, n-1\}$ with $b(i) = 1$ and $b(i+1) = 0$, and the task is to find such an i.

We first give a deterministic protocol using $O(\log^2 n)$ bits. The players maintain integers L, R with $1 \le L < R \le n$ such that $b(L) = 1$ and $b(R) = 0$. Initially $(L, R) = (1, n)$ satisfies the invariant. While $R \ge L + 2$, let $M = \lfloor (L+R)/2 \rfloor$. Alice sends the value $\pi(M)$ to Bob using $\lceil \log_2 n \rceil$ bits. Bob replies with one bit equal to $b(M)$ (that is, whether $\pi(M) \in Y$). If $b(M) = 1$ the players set $L := M$, and if $b(M) = 0$ they set $R := M$. The invariant is preserved because in the first case $b(L) = b(M) = 1$ and $b(R) = 0$, and in the second case $b(L) = 1$ and $b(R) = b(M) = 0$. Moreover $R - L$ decreases by at least a factor of 2 each iteration, so after at most $\lceil \log_2 n \rceil$ iterations one has $R = L + 1$. At that point $b(L) = 1$ and $b(L+1) = 0$, so $i = L$ is a valid output. The communication is at most $\lceil \log_2 n \rceil$ rounds, each round using at most $\lceil \log_2 n \rceil + 1$ bits, hence at most $O(\log^2 n)$ bits total.

We now prove the stated lower bound on monotone formulas for s–t connectivity, where the input is an undirected graph and the output is whether vertex 1 is connected to vertex 2. Let $N = n + 2$ and consider graphs on vertex set $[N]$ with special vertices 1 and 2. Let $m = \binom{N}{2}$ and identify each graph with its edge-indicator vector in $\{0,1\}^m$. Let $f : \{0,1\}^m \to \{0,1\}$ be the monotone Boolean function that outputs 1 if and only if 1 is connected to 2 in the input graph.

Consider the monotone Karchmer–Wigderson communication problem $\mathrm{KW}^m(f)$: Alice receives $x \in f^{-1}(1)$ and Bob receives $y \in f^{-1}(0)$, and they must output an edge-coordinate e such that $x_e = 1$ and $y_e = 0$. Such an e always exists because if $x_e \le y_e$ held for all coordinates then $x \le y$, and by monotonicity $f(x) \le f(y)$ would contradict $f(x) = 1$ and $f(y) = 0$.

We reduce the permutation game to $\mathrm{KW}^m(f)$ as follows. Given Alice's permutation $\pi : [n] \to [n]$, define a graph G_A on vertex set $[N]$ by taking the simple path from 1 to 2 whose internal vertices are ordered by π, namely

$$E(G_A) = \{\{1, \pi(1) + 2\}\} \cup \bigcup_{k=1}^{n-1} \{\{\pi(k) + 2, \pi(k+1) + 2\}\} \cup \{\{\pi(n) + 2, 2\}\}.$$

Then 1 is connected to 2 in G_A, so if x is the edge-indicator vector of G_A then $f(x) = 1$. Given Bob's subset $Y \subseteq [n]$, define a graph G_B on vertex set $[N]$ as the disjoint union of two cliques separated by the cut induced by Y, namely let

$$S = \{1\} \cup \{j + 2 : j \in Y\}, \qquad T = \{2\} \cup \{j + 2 : j \in [n] \setminus Y\},$$

and set $E(G_B) = \{\{u, v\} : u, v \in S,\ u \neq v\} \cup \{\{u, v\} : u, v \in T,\ u \neq v\}$. Then there are no edges between S and T, hence $1 \in S$ and $2 \in T$ are disconnected, so if y is the edge-indicator vector of G_B then $f(y) = 0$. Thus (x, y) is a valid input pair for $\mathrm{KW}^m(f)$.

Assume we have a deterministic protocol Π that solves $\mathrm{KW}^m(f)$ with communication cost C bits. Using the above mapping, Alice and Bob can compute x and y locally from π and Y and then run Π on (x, y), communicating exactly C bits, to obtain an edge $e = \{u, v\}$ with $x_e = 1$ and $y_e = 0$. Since $y_e = 0$ and G_B contains all edges within S and within T, it follows that $\{u, v\}$ crosses the cut (S, T), i.e., one endpoint lies in S and the other lies in T. Since $x_e = 1$ and G_A is a simple path, the edge $\{u, v\}$ must be one of the path edges in G_A. We claim that $\{u, v\}$ cannot be the endpoint edge $\{1, \pi(1) + 2\}$, because by the promise $\pi(1) \in Y$ we have $\pi(1) + 2 \in S$, and $1 \in S$, so $\{1, \pi(1) + 2\}$ is an edge within S and hence belongs to G_B, contradicting $y_e = 0$. Similarly $\{u, v\}$ cannot be the endpoint edge $\{\pi(n) + 2, 2\}$ because $\pi(n) \notin Y$ implies $\pi(n) + 2 \in T$ and $2 \in T$, so that edge lies within T and belongs to G_B. Therefore $\{u, v\}$ must be an internal edge $\{\pi(i) + 2, \pi(i + 1) + 2\}$ for some $i \in \{1, \ldots, n - 1\}$. Since $\{u, v\}$ crosses (S, T), exactly one of $\pi(i) + 2$ and $\pi(i + 1) + 2$ lies in S, hence exactly one of $\pi(i)$ and $\pi(i + 1)$ lies in Y. Because $b(1) = 1$ and $b(n) = 0$, along the path order there must be at least one change from membership to non-membership, and for an internal path edge to cross the cut the only possibility is $b(i) = 1$ and $b(i + 1) = 0$. Consequently the index i satisfies $\pi(i) \in Y$ and $\pi(i + 1) \notin Y$, and Alice can output this i by locating where the returned edge occurs in her path. Thus the permutation game reduces to $\mathrm{KW}^m(f)$ with no additional communication. Since the permutation game requires $\Omega(\log^2 n)$ communication by hypothesis, it follows that any deterministic protocol for $\mathrm{KW}^m(f)$ has communication complexity $\Omega(\log^2 n)$.

We now relate $\mathrm{KW}^m(f)$ to monotone formulas. Let $D(f)$ denote the minimum depth of a monotone formula computing f. We show that $D(f)$ is at least the deterministic communication complexity of $\mathrm{KW}^m(f)$. Let Φ be a monotone formula for f of depth d. We construct a deterministic protocol of cost at most d for $\mathrm{KW}^m(f)$ by following a root-to-leaf path in Φ. On input (x, y) with $f(x) = 1$ and $f(y) = 0$, consider the gate g at the current node, whose output is a monotone function $g(\cdot)$. If g is a leaf labeled by an edge-variable e, then $g(x) = 1$ and $g(y) = 0$ imply $x_e = 1$ and $y_e = 0$, so the protocol outputs e and halts. Otherwise g is either an $\wedge$-gate or a $\vee$-gate with children g_0, g_1. If $g = g_0 \vee g_1$, then $g(x) = 1$ implies $g_0(x) = 1$ or $g_1(x) = 1$; Alice sends one bit indicating an index $j \in \{0, 1\}$ with $g_j(x) = 1$, and the players move to child g_j. Since $g(y) = 0$ implies $g_0(y) = g_1(y) = 0$, it follows that $g_j(y) = 0$ and the invariant $(g_j(x) = 1, g_j(y) = 0)$ holds. If $g = g_0 \wedge g_1$, then $g(y) = 0$ implies $g_0(y) = 0$ or $g_1(y) = 0$; Bob sends one bit indicating an index $j \in \{0, 1\}$ with $g_j(y) = 0$, and the players move to child g_j. Since $g(x) = 1$ implies $g_0(x) = g_1(x) = 1$, we have $g_j(x) = 1$ and again the invariant holds. Each step descends one level and uses one bit, so after at most d bits a leaf is reached and a valid coordinate is output. Hence the deterministic communication complexity of $\mathrm{KW}^m(f)$ is at most $D(f)$, so $D(f) \geq \Omega(\log^2 n)$.

Finally, if a monotone formula has depth d, then its number of leaves is at least 2^d because in a binary formula tree of depth d one has at most 2^d leaves, equivalently a formula with s leaves has depth at most $\lceil \log_2 s \rceil$, so $s \geq 2^d$. Therefore the minimum size $S(f)$ of a monotone formula computing f satisfies

$$S(f) \geq 2^{D(f)} \geq 2^{\Omega(\log^2 n)} = n^{\Omega(\log n)}.$$

This proves that any monotone formula for s–t connectivity must have size at least $n^{\Omega(\log n)}$. □

J.6.3 Claude

Proof. We prove both parts of the theorem.

Part 1: Upper bound protocol with $O(\log^2 n)$ communication.

We describe a protocol using binary search that achieves $O(\log^2 n)$ communication complexity.

The protocol proceeds as follows: Alice and Bob maintain an interval $[L, R]$ with the invariant that $\pi(L) \in Y$ and $\pi(R) \notin Y$. Initially, $L = 1$ and $R = n$.

In each round, they compute the midpoint $M = \lfloor (L + R)/2 \rfloor$ and determine whether $\pi(M) \in Y$. This is done as follows: Bob sends the characteristic vector of Y restricted to a carefully chosen subset, or

equivalently, Alice sends $\pi(M)$ (which requires $\log n$ bits), and Bob responds with one bit indicating whether $\pi(M) \in Y$.

Based on Bob's response:

- If $\pi(M) \in Y$, set $L := M$
- If $\pi(M) \notin Y$, set $R := M$

The interval size decreases by half in each round. After $O(\log n)$ rounds, we have $R = L + 1$, at which point $i = L$ is the desired index.

Each round requires $O(\log n)$ bits of communication (to communicate $\pi(M)$ and Bob's response). With $O(\log n)$ rounds, the total communication is $O(\log^2 n)$.

Part 2: Lower bound for monotone formulas.

Let f be the monotone function on graphs with n vertices that outputs 1 if and only if vertices 1 and 2 are connected. We prove that any monotone formula computing f has size at least $n^{\Omega(\log n)}$.

We use the communication complexity lower bound via the Karchmer-Wigderson framework. For a monotone function f, the Karchmer-Wigderson game is: Alice receives an input x with $f(x) = 1$, Bob receives an input y with $f(y) = 0$, and they must find a coordinate i where $x_i = 1$ and $y_i = 0$.

For our connectivity function, Alice has a graph G_A where 1 and 2 are connected, and Bob has a graph G_B where 1 and 2 are disconnected. They must find an edge that is in G_A but not in G_B.

We reduce the permutation-subset game to this problem. Given a permutation $\pi : [n] \to [n]$ and subset $Y \subseteq [n]$ with $\pi(1) \in Y$ and $\pi(n) \notin Y$:

Alice constructs graph G_A: Create a path $1 = v_0, v_1, \ldots, v_n = 2$ where vertex v_i corresponds to element $\pi(i)$. Add edge (v_i, v_{i+1}) for all i.

Bob constructs graph G_B: Start with the same vertex set. Add edge (v_i, v_{i+1}) only if both $\pi(i) \in Y$ and $\pi(i+1) \in Y$, or both $\pi(i) \notin Y$ and $\pi(i+1) \notin Y$.

In G_A, vertices 1 and 2 are connected by the path. In G_B, there is no path from 1 to 2 because any path would need to cross from Y to its complement, but such edges are not present in G_B.

Any edge (v_i, v_{i+1}) in G_A but not in G_B corresponds to an index i where $\pi(i) \in Y$ and $\pi(i+1) \notin Y$, which solves the permutation-subset game.

Since the permutation-subset game requires $\Omega(\log^2 n)$ communication, the Karchmer-Wigderson game for the connectivity function requires $\Omega(\log^2 n)$ communication.

By the Karchmer-Wigderson relation, the depth of any monotone formula for f is at least $\Omega(\log^2 n)$. Since the connectivity function on n vertices can be computed by a monotone formula of depth d with size at most $\binom{n}{2}^d$ (there are at most $\binom{n}{2}$ edges), and we need depth at least $c \log^2 n$ for some constant $c > 0$, we obtain:

$$\text{Size} \geq 2^d \geq 2^{c \log^2 n} = n^{c \log n} = n^{\Omega(\log n)}$$

This completes the proof. □

J.6.4 Gemini

Proof. **Part 1: Protocol with $O(\log^2 n)$ communication**

Let Alice hold the permutation $\pi : [n] \to [n]$ and Bob hold the subset $Y \subseteq [n]$. The promise is that $\pi(1) \in Y$ and $\pi(n) \notin Y$. They wish to find an index i such that $\pi(i) \in Y$ and $\pi(i+1) \notin Y$.

The players execute a binary search on the indices of the permutation.

1. Both players maintain a search interval $[L, R] \subseteq [1, n]$. Initially, $L = 1$ and $R = n$. The invariant maintained is $\pi(L) \in Y$ and $\pi(R) \notin Y$.
2. While $R > L + 1$:
   (a) The players compute the midpoint $M = \lfloor \frac{L+R}{2} \rfloor$.
   (b) Alice sends the value $v = \pi(M)$ to Bob. Since $v \in [n]$, this requires $\lceil \log n \rceil$ bits.

(c) Bob checks if $v \in Y$. He sends a single bit b to Alice, where $b = 1$ if $v \in Y$ and $b = 0$ if $v \notin Y$.

(d) If $b = 1$ (i.e., $\pi(M) \in Y$), both players update $L \leftarrow M$. The invariant holds since $\pi(M) \in Y$ and $\pi(R) \notin Y$.

(e) If $b = 0$ (i.e., $\pi(M) \notin Y$), both players update $R \leftarrow M$. The invariant holds since $\pi(L) \in Y$ and $\pi(M) \notin Y$.

3. When $R = L + 1$, the loop terminates. By the invariant, $\pi(L) \in Y$ and $\pi(L + 1) \notin Y$. The players output $i = L$.

Complexity Analysis: The size of the interval $R - L$ is initially $n - 1$ and is halved in each iteration. The number of rounds is $O(\log n)$. In each round, Alice sends $O(\log n)$ bits and Bob sends 1 bit. The total communication complexity is $O(\log n) \cdot O(\log n) = O(\log^2 n)$.

Part 2: Lower bound for monotone graph connectivity

Let f be the monotone function corresponding to st-connectivity on a graph with N vertices (where $N \approx n$). Specifically, let the vertices be $V = \{1, \ldots, n\}$. The input to f is an undirected graph $G = (V, E)$, represented by variables x_{uv} for each potential edge. $f(G) = 1$ if and only if vertex 1 is connected to vertex 2.

We utilize the Karchmer-Wigderson relation. For a monotone function f, the depth of the minimal monotone formula computing f, denoted $d(f)$, is equal to the communication complexity of the Karchmer-Wigderson game KW_f. In the game KW_f:

- Alice receives a minterm of f, which corresponds to a minimal graph G_A where 1 is connected to 2. This is a simple path from 1 to 2.
- Bob receives a maxterm of f, which corresponds to a maximal graph G_B where 1 is not connected to 2. This corresponds to a partition of vertices $(S, V \setminus S)$ with $1 \in S$ and $2 \notin S$, where G_B contains all edges except those crossing the cut between S and $V \setminus S$.
- Their goal is to find an edge e such that $e \in G_A$ and $e \notin G_B$.

We reduce the Permutation Game to KW_f. Let (π, Y) be an instance of the Permutation Game on $[n]$ with the promise $\pi(1) \in Y$ and $\pi(n) \notin Y$. To align the boundary conditions with the st-connectivity problem (connecting 1 to 2), we can augment the permutation game or simply embed it. Consider the graph vertices to be the set of values in the permutation. Without loss of generality for the asymptotic lower bound, assume the permutation game is hard even when restricted to $\pi(1) = 1$ and $\pi(n) = 2$ (or we map the values $\pi(1)$ to node 1 and $\pi(n)$ to node 2).

Reduction:

1. Alice constructs a graph G_A consisting of the path $P = (\pi(1), \pi(2), \ldots, \pi(n))$. The edges are $e_j = \{\pi(j), \pi(j + 1)\}$ for $j = 1, \ldots, n - 1$. Since $\pi(1) = 1$ and $\pi(n) = 2$, $f(G_A) = 1$.
2. Bob constructs a cut based on Y. Let $S = Y$. Since $1 \in Y$ and $2 \notin Y$, S is a valid cut separating 1 and 2. Bob's graph G_B contains all edges $\{u, v\}$ such that $\{u, v\} \subseteq S$ or $\{u, v\} \subseteq V \setminus S$. $f(G_B) = 0$.
3. Alice and Bob run the protocol for KW_f. The output is an edge $e = \{\pi(i), \pi(i + 1)\}$ such that $e \in G_A$ and $e \notin G_B$.
4. Since $e \in G_A$, it is an edge on the path. Since $e \notin G_B$, the edge must cross the cut defined by S. Thus, one endpoint is in S and the other is not.
5. Given the direction of the path from $1 \in S$ to $2 \notin S$, there must exist an index i such that $\pi(i) \in S$ and $\pi(i + 1) \notin S$. The edge returned by the KW protocol identifies such an i.

6. Since $S = Y$, this index i satisfies $\pi(i) \in Y$ and $\pi(i+1) \notin Y$, solving the Permutation Game.

Since the Permutation Game requires $\Omega(\log^2 n)$ communication, we have:

$$D(KW_f) \geq \Omega(\log^2 n)$$

By the Karchmer-Wigderson theorem, the depth of any monotone formula for st-connectivity is $d(f) = \Omega(\log^2 n)$. The size of a formula $L(f)$ is related to its depth by $L(f) \geq 2^{d(f)}$. Therefore:

$$L(f) \geq 2^{\Omega(\log^2 n)} = n^{\Omega(\log n)}$$

Thus, any monotone formula for st-connectivity must have size $n^{\Omega(\log n)}$. □

K Problems in Chapter 10

K.1 Problem 10.1

K.1.1 Problem Statement

Theorem K.1. *Any randomized algorithm that estimates the ∞-moment $\max_i f(i)$ within an additive factor of 1 must use memory $\Omega(n)$.*

K.1.2 GPT

Proof. Let $n \in \mathbb{N}$ and consider the one-way randomized communication problem INDEX_n: Alice receives $x \in \{0,1\}^n$, Bob receives $j \in [n] = \{1, \ldots, n\}$, and Bob must output x_j with error probability at most $1/3$.

We first prove that any one-way randomized protocol for INDEX_n with error at most $1/3$ must send $\Omega(n)$ bits. Assume x is uniformly distributed on $\{0,1\}^n$. Let the protocol use private randomness for Alice and Bob. Let M denote the random message sent by Alice; thus M is a (possibly randomized) function of x, and hence $H(M) \leq \ell$ where ℓ is the maximum message length in bits. For each $k \in [n]$, let $\widehat{x}_k$ be Bob's output when his index is k; since the protocol has error at most $1/3$ for every input pair, we have $\Pr[\widehat{x}_k = x_k] \geq 2/3$ for every k (over the joint distribution of x and the private randomness). Since x_k is uniformly distributed in $\{0,1\}$ and $\widehat{x}_k$ is a function of (M, k) and Bob's private randomness, we have

$$H(x_k \mid M) \leq H(x_k \mid M, k) \leq H(x_k \mid \widehat{x}_k) \leq h(1/3),$$

where $h(p) = -p \log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function, and the last inequality follows because the conditional distribution of x_k given $\widehat{x}_k$ has error probability at most $1/3$. Therefore

$$I(x_k; M) = H(x_k) - H(x_k \mid M) \geq 1 - h(1/3)$$

for every $k \in [n]$. Using the chain rule for entropy and the fact that conditioning reduces entropy, we have

$$H(x \mid M) = H(x_1, \ldots, x_n \mid M) \leq \sum_{k=1}^{n} H(x_k \mid M),$$

and thus

$$I(x; M) = H(x) - H(x \mid M) \geq n - \sum_{k=1}^{n} H(x_k \mid M) = \sum_{k=1}^{n} (1 - H(x_k \mid M)) = \sum_{k=1}^{n} I(x_k; M) \geq n(1 - h(1/3)).$$

Since mutual information is bounded by the entropy of the message,

$$I(x; M) \leq H(M) \leq \ell,$$

we conclude $\ell \geq n(1 - h(1/3))$, which is $\Omega(n)$.

We now reduce INDEX_n to the streaming problem of estimating $F_\infty = \max_i f(i)$ to additive error 1. Suppose there exists a randomized one-pass streaming algorithm $\mathcal{A}$ which, for every frequency vector $f : [n] \to \mathbb{Z}_{\geq 0}$ generated by an insertion-only stream over universe $[n]$, outputs an estimate $\widetilde{F}$ such that

$$\Pr\big[\, |\widetilde{F} - \max_{i\in[n]} f(i)| \leq 1 \,\big] \geq 2/3,$$

and suppose $\mathcal{A}$ uses at most s bits of memory.

Given an input (x, j) to INDEX_n, Alice simulates $\mathcal{A}$ on the insertion stream that, for each $i \in [n]$ with $x_i = 1$, inserts item i exactly twice (and inserts nothing for indices with $x_i = 0$). After this stream, the resulting frequency vector f satisfies $f(i) \in \{0, 2\}$ for all i. Alice sends Bob the memory contents (the internal state) of $\mathcal{A}$ after processing this stream; this message has length at most s bits.

Bob continues the simulation of $\mathcal{A}$ from the received state and performs four additional insertions of item j. Let f' be the final frequency vector. Then $f'(j) = f(j) + 4 = 4 + 2x_j$, and for every $i \neq j$ we have $f'(i) = f(i) \in \{0, 2\}$. Consequently,

$$\max_{i\in[n]} f'(i) = \begin{cases} 4 & \text{if } x_j = 0, \\ 6 & \text{if } x_j = 1. \end{cases}$$

Bob runs $\mathcal{A}$ to obtain $\widetilde{F}$ and outputs 1 if $\widetilde{F} > 5$ and outputs 0 otherwise. On the event that $|\widetilde{F} - \max_i f'(i)| \leq 1$, we have: if $x_j = 0$ then $\max_i f'(i) = 4$ so $\widetilde{F} \leq 5$ and Bob outputs 0; if $x_j = 1$ then $\max_i f'(i) = 6$ so $\widetilde{F} \geq 5$ and in fact $\widetilde{F} > 5$ unless $\widetilde{F} = 5$, but $\widetilde{F} \geq 5$ and the strict threshold > 5 yields correctness whenever $\widetilde{F} \in [5, 7]$ and $\widetilde{F} \neq 5$. To eliminate the boundary, Bob instead uses the threshold $11/2$: he outputs 1 if $\widetilde{F} > 11/2$ and outputs 0 otherwise. Then on the event $|\widetilde{F} - 4| \leq 1$ we have $\widetilde{F} \leq 5 < 11/2$, and on the event $|\widetilde{F} - 6| \leq 1$ we have $\widetilde{F} \geq 5 > 11/2$. Hence Bob outputs x_j correctly whenever the additive-error guarantee holds. Therefore this defines a one-way randomized protocol for INDEX_n with message length at most s and error probability at most $1/3$.

By the lower bound proved above, any such one-way protocol must have message length $\Omega(n)$, hence $s = \Omega(n)$. This proves that any randomized streaming algorithm estimating $\max_i f(i)$ within additive factor 1 must use memory $\Omega(n)$. □

K.1.3 Claude

Proof. We prove this lower bound using Yao's minimax principle by constructing a hard distribution over inputs and showing that any deterministic algorithm must fail on a constant fraction of inputs from this distribution unless it uses $\Omega(n)$ memory.

Let n be the size of the domain. We construct a distribution $\mathcal{D}$ over frequency vectors as follows: Choose uniformly at random an index $j^* \in [n]$, and set $f(j^*) = 2$ and $f(i) = 1$ for all $i \neq j^*$. Thus, the true ∞-moment is $\max_i f(i) = 2$.

Consider any deterministic streaming algorithm $\mathcal{A}$ that uses s bits of memory. The algorithm processes a stream that presents the elements in some order. We construct the stream adversarially: present $n - 1$ distinct elements each appearing once, then present one additional element.

For any deterministic algorithm with memory $s < \log_2 n$ bits, there are at most $2^s < n$ possible memory states after seeing the first $n - 1$ distinct elements. By the pigeonhole principle, there exist at least two distinct sets $S_1, S_2 \subseteq [n]$ with $|S_1| = |S_2| = n - 1$ such that the algorithm reaches the same memory state M after processing streams consisting of the elements in S_1 and S_2 respectively (each element appearing once).

Let $j_1 \in [n] \setminus S_1$ and $j_2 \in [n] \setminus S_2$ be the unique elements not in S_1 and S_2 respectively. Since $S_1 \neq S_2$ and both have size $n - 1$, we can assume without loss of generality that $j_1 \neq j_2$.

Now consider two possible continuations from state M:

Case 1: The next element is j_1. If $j^* = j_1$, then $f(j_1) = 2$ and $\max_i f(i) = 2$. Otherwise, $f(j_1) = 1$ and $\max_i f(i) = 2$ (since $j^* \in S_1$).

Case 2: The next element is j_2. If $j^* = j_2$, then $f(j_2) = 2$ and $\max_i f(i) = 2$. Otherwise, $f(j_2) = 1$ and $\max_i f(i) = 2$ (since $j^* \in S_2$).

However, the algorithm cannot distinguish between these cases when starting from the same state M. Under our distribution $\mathcal{D}$, with probability $\frac{1}{n}$, we have $j^* = j_1$, and with probability $\frac{1}{n}$, we have $j^* = j_2$.

More precisely, consider the scenario where after processing S_1, the element j_1 appears again. The algorithm must output an estimate $\widehat{m}$ of $\max_i f(i)$. If $j^* \in S_1$, then some element in S_1 appeared twice in the full stream, so $\max_i f(i) = 2$. If $j^* = j_1$, then $f(j_1) = 2$ and $\max_i f(i) = 2$.

The key observation is that for $s < \log_2 n - O(1)$, the number of distinguishable memory states is insufficient to remember which specific element among the $n - 1$ elements seen has frequency 2. Therefore, when queried, the algorithm cannot determine whether the maximum frequency is 1 or 2 with error less than 1 for a constant fraction of inputs.

By Yao's minimax principle, if every deterministic algorithm using s bits fails with probability at least $\Omega(1)$ on distribution $\mathcal{D}$, then any randomized algorithm must use $\Omega(n)$ bits to succeed with high probability.

Since $s = o(n)$ implies $s < \log_2 n$ for sufficiently large n, and the algorithm cannot track which element has frequency 2 without storing $\Omega(n)$ bits of information, we conclude that any randomized algorithm estimating the ∞-moment within additive error 1 requires $\Omega(n)$ memory. □

K.1.4 Gemini

Proof. We prove this lower bound via a reduction from the Set Disjointness problem in two-party communication complexity. Let DISJ_n denote the problem where Alice holds a characteristic vector $x \in \{0,1\}^n$ and Bob holds a characteristic vector $y \in \{0,1\}^n$, corresponding to subsets $S_A, S_B \subseteq \{1, \ldots, n\}$. Their goal is to determine whether $S_A \cap S_B = \emptyset$, which is equivalent to checking if $\sum_{i=1}^n x_i y_i = 0$. It is a celebrated result in communication complexity that the randomized one-way communication complexity of DISJ_n is $\Omega(n)$.

Assume there exists a randomized streaming algorithm $\mathcal{A}$ that estimates $F_\infty = \max_i f_i$ within an additive factor of 1 with probability at least $2/3$. We construct a protocol for DISJ_n using $\mathcal{A}$ as follows:

1. Alice initializes an instance of algorithm $\mathcal{A}$. 2. For each index $i \in \{1, \ldots, n\}$ such that $x_i = 1$, Alice inserts the item i into the stream exactly 3 times. 3. Alice transmits the memory state of $\mathcal{A}$ to Bob. 4. Bob continues the execution of $\mathcal{A}$ starting from the received memory state. 5. For each index $i \in \{1, \ldots, n\}$ such that $y_i = 1$, Bob inserts the item i into the stream exactly 3 times. 6. Bob queries $\mathcal{A}$ for the estimate $\widehat{F}_\infty$.

Let f_i denote the final frequency of item i in the stream. The frequency is given by:

$$f_i = 3x_i + 3y_i.$$

We analyze the value of the infinite moment $F_\infty = \max_i f_i$ in the two cases of the disjointness problem:

Case 1: The sets are disjoint ($S_A \cap S_B = \emptyset$). In this case, for all i, it is impossible that both $x_i = 1$ and $y_i = 1$. Thus, $x_i + y_i \in \{0, 1\}$. Consequently, $f_i \in \{0, 3\}$ for all i.

$$F_\infty = \max_i f_i \leq 3.$$

Since $\mathcal{A}$ provides an estimate $\widehat{F}_\infty$ within an additive factor of 1, we have:

$$|\widehat{F}_\infty - F_\infty| \leq 1 \implies \widehat{F}_\infty \leq F_\infty + 1 \leq 4.$$

Case 2: The sets are not disjoint ($S_A \cap S_B \neq \emptyset$). In this case, there exists at least one index j such that $x_j = 1$ and $y_j = 1$. For this index, $f_j = 3(1) + 3(1) = 6$.

$$F_\infty = \max_i f_i \geq 6.$$

Using the accuracy guarantee of $\mathcal{A}$:

$$|\widehat{F}_\infty - F_\infty| \leq 1 \implies \widehat{F}_\infty \geq F_\infty - 1 \geq 5.$$

Bob can distinguish between the two cases by checking if $\widehat{F}_\infty \geq 4.5$. If $\widehat{F}_\infty < 4.5$, the sets are disjoint; otherwise, they intersect. This protocol solves DISJ_n with the same success probability as $\mathcal{A}$.

Since any protocol solving DISJ_n requires $\Omega(n)$ bits of communication, the size of the memory state transmitted from Alice to Bob must be $\Omega(n)$. Therefore, any randomized algorithm estimating F_∞ within an additive factor of 1 requires $\Omega(n)$ space. □

K.2 Problem 10.2

K.2.1 Problem Statement

Theorem K.2. *Consider the connectivity problem for streaming algorithms. The input stream consists of edges in a graph with vertex set $[n]$. The algorithm should decide if the vertices 1 and n are connected by a path.*

1. This problem can be solved with $O(n \log n)$ memory.

2. The memory size of such an algorithm is at least $\Omega(n)$.

Hint: Use such an algorithm to solve disjointness.

K.2.2 GPT

Proof. We first give a deterministic one-pass streaming algorithm using $O(n \log n)$ bits. Maintain a disjoint-set union (Union-Find) data structure over the vertex set $[n]$. For each $v \in [n]$, store a parent pointer $p(v) \in [n]$ and a rank (or size) value $r(v) \in \{0, 1, \ldots, \lfloor \log_2 n \rfloor\}$. Initialize $p(v) = v$ and $r(v) = 0$ for all v. Upon receiving an edge (u, v) in the stream, perform $\mathrm{union}(u, v)$ by finding the representatives $\mathrm{find}(u)$ and $\mathrm{find}(v)$ and linking the smaller-rank root to the larger-rank root, breaking ties by increasing rank by 1. After processing the stream, output YES if and only if $\mathrm{find}(1) = \mathrm{find}(n)$. Correctness holds because the Union-Find invariants imply that after all unions, two vertices have the same representative if and only if they are connected in the graph formed by all streamed edges: each edge causes its endpoints' components to merge, hence the maintained partition coincides with the connected components of the processed graph. The memory usage is $O(n \log n)$ bits because each parent pointer $p(v)$ requires $\lceil \log_2 n \rceil$ bits and each rank requires $O(\log n)$ bits, so the total is $O(n \log n)$ bits.

We now prove a lower bound of $\Omega(n)$ bits on the memory of any (randomized or deterministic) one-pass streaming algorithm that decides whether 1 and n are connected, with constant error probability allowed. We reduce from the two-party communication problem DISJ_m for $m = n - 1$, where Alice has $x \in \{0, 1\}^m$, Bob has $y \in \{0, 1\}^m$, and they must decide whether there exists an index $i \in [m]$ with $x_i = y_i = 1$. It is known that the randomized communication complexity of DISJ_m with error at most $1/3$ is $\Omega(m)$ bits, and the deterministic communication complexity is $\Omega(m)$ bits as well.

Fix $n \geq 2$ and let $m = n - 1$. Given inputs $x, y \in \{0, 1\}^m$, define a graph on vertex set $[n]$ by including, for each $i \in [m]$, the edge $(1, i + 1)$ if $x_i = 1$, and the edge $(i + 1, n)$ if $y_i = 1$. Denote this graph by $G(x, y)$. Observe that 1 is connected to n in $G(x, y)$ if and only if there exists $i \in [m]$ such that both edges $(1, i + 1)$ and $(i + 1, n)$ are present, which holds if and only if there exists i with $x_i = y_i = 1$. Indeed, any path from 1 to n must leave 1 via an edge of the form $(1, i + 1)$ and enter n via an edge of the form $(i + 1, n)$, and since the only edges incident to vertices $2, \ldots, n - 1$ are those to 1 and to n, the only possible 1–n paths have length 2 and are of the form $1 \to (i + 1) \to n$. Thus

$$(1 \text{ is connected to } n \text{ in } G(x, y)) \iff \bigvee_{i=1}^{m} (x_i \wedge y_i).$$

Let $\mathcal{A}$ be a one-pass streaming algorithm for the connectivity problem on $[n]$ that uses at most S bits of memory and errs with probability at most $1/3$ over its internal randomness (in the deterministic case, the error probability is 0). We construct a two-party communication protocol for DISJ_m with communication at most S bits as follows. Alice simulates $\mathcal{A}$ on the first part of the stream consisting of all edges $(1, i + 1)$ with $x_i = 1$, in any fixed order, starting from the initial state of $\mathcal{A}$. Let the resulting memory state be σ. Alice sends σ to Bob using at most S bits. Bob continues the simulation of $\mathcal{A}$ starting from state σ on the remaining edges $(i + 1, n)$ with $y_i = 1$, in any fixed order, and outputs whatever $\mathcal{A}$ outputs. The concatenation of the two edge subsequences is exactly a stream describing $G(x, y)$, so by correctness of $\mathcal{A}$ we have that Bob's output equals the connectivity predicate of $G(x, y)$, which equals $\neg\mathrm{DISJ}_m(x, y)$, with the same error probability as $\mathcal{A}$. Therefore, this yields a communication protocol for DISJ_m (or for

its negation, which has the same communication complexity) with communication at most S bits and error at most $1/3$.

By the lower bound on the (randomized or deterministic) communication complexity of DISJ_m, it follows that $S = \Omega(m) = \Omega(n)$. Hence any one-pass streaming algorithm for deciding whether 1 and n are connected must use at least $\Omega(n)$ bits of memory. Combining this with the $O(n \log n)$-bit upper bound completes the proof. □

K.2.3 Claude

Proof. We prove both parts of the theorem.

Part 1: Upper bound of $O(n \log n)$ memory.

We use a union-find data structure to maintain the connected components of the graph as edges arrive in the stream. The algorithm maintains a partition of the vertex set $[n]$ into disjoint sets, where each set represents a connected component.

For each edge (u, v) in the stream, we perform a union operation that merges the components containing u and v. The union-find data structure with path compression and union by rank requires $O(n)$ space to store the parent pointers and ranks for n vertices. Each operation (find and union) takes $O(\alpha(n))$ amortized time, where α is the inverse Ackermann function, which is effectively constant.

To store vertex identifiers and auxiliary information, we need $O(\log n)$ bits per vertex, giving total space $O(n \log n)$ bits. After processing all edges, we check whether vertices 1 and n belong to the same component by performing two find operations.

Therefore, the connectivity problem can be solved with $O(n \log n)$ memory.

Part 2: Lower bound of $\Omega(n)$ memory.

We prove this lower bound by reduction from the two-party communication complexity problem of set disjointness (DISJ). In DISJ, Alice holds a subset $A \subseteq [m]$ and Bob holds a subset $B \subseteq [m]$, and they must determine whether $A \cap B = \emptyset$. It is well-known that the randomized communication complexity of DISJ is $\Omega(m)$ bits.

Given an instance of DISJ with sets $A, B \subseteq [m]$, we construct a connectivity instance on a graph with $n = m + 2$ vertices labeled $\{0, 1, 2, \ldots, m, m + 1\}$. We identify vertex 0 with vertex 1 in the theorem statement and vertex $m + 1$ with vertex n.

Alice creates edges from vertex 0 to vertex i for each $i \in A$, where $1 \leq i \leq m$. Bob creates edges from vertex i to vertex $m + 1$ for each $i \in B$, where $1 \leq i \leq m$.

The key observation is that vertices 0 and $m + 1$ are connected in the resulting graph if and only if $A \cap B \neq \emptyset$. Indeed, any path from 0 to $m + 1$ must pass through some intermediate vertex $i \in [m]$. Such a path exists if and only if there exists i such that $(0, i)$ is an edge (meaning $i \in A$) and $(i, m + 1)$ is an edge (meaning $i \in B$), which occurs if and only if $i \in A \cap B$.

Now suppose there exists a streaming algorithm for connectivity using s bits of memory. We can use this algorithm to solve DISJ as follows: Alice simulates the algorithm on her edges, obtaining a memory state of s bits. She sends this state to Bob, who continues the simulation with his edges and outputs the connectivity result. This gives a one-way communication protocol for DISJ using s bits of communication.

Since the randomized communication complexity of DISJ on m-element sets is $\Omega(m)$, we have $s = \Omega(m)$. With $n = m + 2$, this gives $s = \Omega(n)$.

Therefore, any streaming algorithm for connectivity requires $\Omega(n)$ bits of memory. □

K.2.4 Gemini

Proof. We prove the two parts of the theorem sequentially.

Part 1: Upper Bound. We demonstrate that the connectivity problem can be solved using $O(n \log n)$ bits of memory. We employ a Disjoint Set Union (DSU) data structure to maintain the connected components of the graph. Let the vertex set be $V = \{1, \ldots, n\}$. The algorithm maintains an array P of size n, where $P[i]$ represents the parent of vertex i. Initially, $P[i] = i$ for all $i \in \{1, \ldots, n\}$, indicating that each vertex forms a singleton component.

For each edge (u, v) arriving in the stream, the algorithm performs the `Union` operation. We find the representatives (roots) of the components containing u and v by following the parent pointers until a

vertex r with $P[r] = r$ is reached. Let r_u and r_v be the roots of u and v respectively. If $r_u \neq r_v$, we update the parent array by setting $P[r_u] = r_v$.

After the stream is exhausted, the algorithm computes the roots of vertices 1 and n. If $r_1 = r_n$, the algorithm outputs that 1 and n are connected; otherwise, it outputs that they are not.

The correctness follows from the standard properties of the DSU data structure, which correctly maintains partition of vertices into connected components. Regarding space complexity, the algorithm stores the array P of length n. Each entry $P[i]$ stores an integer index in $\{1, \ldots, n\}$, which requires $\lceil \log_2 n \rceil$ bits. Thus, the total memory usage is $n\lceil \log_2 n \rceil$ bits, which is $O(n \log n)$.

Part 2: Lower Bound. We prove the lower bound via a reduction from the Set Disjointness problem (DISJ) in the one-way communication complexity model. In the DISJ problem, Alice holds a characteristic vector $x \in \{0,1\}^k$ and Bob holds a characteristic vector $y \in \{0,1\}^k$. They must determine if there exists an index i such that $x_i = 1$ and $y_i = 1$. It is a standard result in communication complexity that any randomized protocol solving DISJ with probability at least $2/3$ requires $\Omega(k)$ bits of communication.

Assume there exists a streaming algorithm $\mathcal{A}$ that solves the 1-n connectivity problem on a graph with n vertices using $S(n)$ bits of memory. We construct a protocol for DISJ with input size $k = n - 2$. Let the graph vertices be labeled $\{1, \ldots, n\}$. We associate the indices $\{1, \ldots, k\}$ of the DISJ instance with the intermediate vertices $\{2, \ldots, n-1\}$.

Alice processes her input x and feeds edges into the algorithm $\mathcal{A}$ as follows: for each $i \in \{1, \ldots, k\}$, if $x_i = 1$, Alice adds the edge $(1, i+1)$ to the stream. After processing all such edges, Alice transmits the memory state of $\mathcal{A}$ to Bob. The size of this message is $S(n)$ bits.

Bob initializes $\mathcal{A}$ with the received memory state. He processes his input y as follows: for each $i \in \{1, \ldots, k\}$, if $y_i = 1$, Bob adds the edge $(i+1, n)$ to the stream. After processing these edges, Bob queries $\mathcal{A}$ to check if vertices 1 and n are connected.

Consider the structure of the graph constructed by the union of Alice's and Bob's edges. Any path from 1 to n must pass through an intermediate vertex $v \in \{2, \ldots, n-1\}$. Specifically, a path exists if and only if there is some $j \in \{2, \ldots, n-1\}$ such that the edge $(1, j)$ exists and the edge (j, n) exists. By construction, $(1, i+1)$ exists if and only if $x_i = 1$, and $(i+1, n)$ exists if and only if $y_i = 1$. Therefore, vertices 1 and n are connected if and only if there exists an i such that $x_i = 1$ and $y_i = 1$, which corresponds to the sets not being disjoint.

Since the protocol solves DISJ using $S(n)$ bits of communication, and the communication complexity of DISJ is $\Omega(k)$, we have:

$$S(n) \geq \Omega(k) = \Omega(n-2) = \Omega(n).$$

Thus, the memory size of the streaming algorithm is at least $\Omega(n)$. □

K.3 Problem 10.3

K.3.1 Problem Statement

Theorem K.3. *In this exercise, we prove Barrington's theorem. For any fixed input, an interval in a width 5 program can be thought of as computing a map from* $[5]$ *to* $[5]$.

1. Show that there are two permutations $\pi, \sigma : [5] \to [5]$ with the property that $\pi\sigma \neq \sigma\pi$. Here $\pi\sigma$ denotes the map obtained by composing π and σ.

2. Prove by induction on d that if there is a Boolean circuit computing $f(x_1, \ldots, x_n)$ with depth d, then there are three width 5 programs, each of length $O(4^d)$, computing each of the maps $\pi^{f(x_1,\ldots,x_n)}$, $\pi^{-f(x_1,\ldots,x_n)}$, $\sigma^{f(x_1,\ldots,x_n)}$. Here $\pi^1 = \pi$, and π^0 is the identity.

3. Given a Boolean circuit of depth d, show that there is a branching program of width 5 and length $O(4^d)$ that computes the same function.

K.3.2 GPT

Proof. We prove the theorem in three parts corresponding to the three statements in the problem.

Part 1: Existence of two non-commuting permutations

Let us define the two permutations $\pi, \sigma : [5] \to [5]$ explicitly.

Let π be the permutation of $[5]$ given by

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix},$$

which is a cyclic permutation shifting each element to the right by one position.

Next, define σ as the permutation

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 1 & 2 \end{pmatrix},$$

which is another cyclic permutation but shifting each element to the right by two positions.

We now verify that $\pi\sigma \neq \sigma\pi$. First, compute the composition $\pi\sigma$. For each $i \in [5]$, we apply σ first and then π:

$$\pi\sigma(1) = \pi(3) = 4, \quad \pi\sigma(2) = \pi(4) = 5, \quad \pi\sigma(3) = \pi(5) = 1, \quad \pi\sigma(4) = \pi(1) = 2, \quad \pi\sigma(5) = \pi(2) = 3.$$

Thus, we have

$$\pi\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 1 & 2 & 3 \end{pmatrix}.$$

Now, compute the composition $\sigma\pi$. For each $i \in [5]$, we apply π first and then σ:

$$\sigma\pi(1) = \sigma(2) = 4, \quad \sigma\pi(2) = \sigma(3) = 5, \quad \sigma\pi(3) = \sigma(4) = 1, \quad \sigma\pi(4) = \sigma(5) = 2, \quad \sigma\pi(5) = \sigma(1) = 3.$$

Thus, we have

$$\sigma\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 1 & 2 & 3 \end{pmatrix}.$$

Clearly, $\pi\sigma \neq \sigma\pi$ as the two compositions result in different mappings. Therefore, the permutations π and σ do not commute.

Part 2: Induction on depth for width 5 programs

We prove the statement by induction on d.

Base case: $d = 0$

For $d = 0$, the Boolean circuit is a constant function, so the function $f(x_1, \ldots, x_n)$ does not depend on any input variables. Hence, $f(x_1, \ldots, x_n) = 0$ or $f(x_1, \ldots, x_n) = 1$. In this case, we can choose the identity permutation for $\pi^{f(x_1,\ldots,x_n)}$ and $\pi^{-f(x_1,\ldots,x_n)}$, and $\sigma^{f(x_1,\ldots,x_n)}$ as follows:

$$\pi^0 = \pi, \quad \pi^1 = \pi, \quad \sigma^0 = \sigma, \quad \sigma^1 = \sigma.$$

Thus, we have three width 5 programs of length $O(1)$ computing each of these maps, as the depth of the Boolean circuit is zero.

Inductive step: Assume the statement holds for depth d

Suppose that for a Boolean circuit of depth d, there exist three width 5 programs, each of length $O(4^d)$, computing the maps $\pi^{f(x_1,\ldots,x_n)}, \pi^{-f(x_1,\ldots,x_n)}, \sigma^{f(x_1,\ldots,x_n)}$.

Now, consider a Boolean circuit of depth $d + 1$. A circuit of depth $d + 1$ can be decomposed into a circuit of depth d followed by a single gate. We can express the output of the circuit as $f(x_1, \ldots, x_n) = g(h(x_1, \ldots, x_n), x_{n+1})$, where g is the final gate and $h(x_1, \ldots, x_n)$ represents the output of the sub-circuit of depth d.

By the induction hypothesis, for the sub-circuit computing $h(x_1, \ldots, x_n)$, there are three width 5 programs of length $O(4^d)$ computing $\pi^{h(x_1,\ldots,x_n)}, \pi^{-h(x_1,\ldots,x_n)}, \sigma^{h(x_1,\ldots,x_n)}$. Applying the gate g to these maps, we can construct three width 5 programs of length $O(4^{d+1})$ that compute $\pi^{f(x_1,\ldots,x_n)}, \pi^{-f(x_1,\ldots,x_n)}, \sigma^{f(x_1,\ldots,x_n)}$. This completes the induction step.

Thus, by induction, we have shown that for any Boolean circuit of depth d, there are three width 5 programs, each of length $O(4^d)$, computing the required maps.

Part 3: Boolean circuit to branching program

Given a Boolean circuit of depth d, we aim to show that there exists a branching program of width 5 and length $O(4^d)$ that computes the same function.

By the previous part, for the given Boolean circuit, there are three width 5 programs of length $O(4^d)$ computing the maps $\pi^{f(x_1,\ldots,x_n)}$, $\pi^{-f(x_1,\ldots,x_n)}$, $\sigma^{f(x_1,\ldots,x_n)}$. These programs can be interpreted as branching programs where each step corresponds to the application of a permutation π or σ, depending on the value of the input bits. Since the number of steps in each program is bounded by $O(4^d)$, the total length of the branching program is also $O(4^d)$.

Therefore, we can construct a branching program of width 5 and length $O(4^d)$ that computes the same function as the given Boolean circuit. This completes the proof.

□

K.3.3 Claude

Proof. We prove each part of Barrington's theorem in turn.

Part 1. We must show that there exist two permutations $\pi, \sigma : [5] \to [5]$ such that $\pi\sigma \neq \sigma\pi$.

Consider the permutations of $[5] = \{1, 2, 3, 4, 5\}$ given by:

$$\pi = (1\ 2\ 3\ 4\ 5) \quad \text{and} \quad \sigma = (1\ 2)$$

where $(1\ 2\ 3\ 4\ 5)$ denotes the 5-cycle mapping $1 \mapsto 2 \mapsto 3 \mapsto 4 \mapsto 5 \mapsto 1$, and $(1\ 2)$ denotes the transposition swapping 1 and 2.

Computing $\pi\sigma$: First apply σ, then π. We have $\sigma(1) = 2$ and $\pi(2) = 3$, so $(\pi\sigma)(1) = 3$.

Computing $\sigma\pi$: First apply π, then σ. We have $\pi(1) = 2$ and $\sigma(2) = 1$, so $(\sigma\pi)(1) = 1$.

Since $(\pi\sigma)(1) = 3 \neq 1 = (\sigma\pi)(1)$, we conclude that $\pi\sigma \neq \sigma\pi$.

Part 2. We prove by induction on d that if there is a Boolean circuit computing $f(x_1, \ldots, x_n)$ with depth d, then there are three width 5 programs, each of length $O(4^d)$, computing the maps $\pi^{f(x_1,\ldots,x_n)}$, $\pi^{-f(x_1,\ldots,x_n)}$, and $\sigma^{f(x_1,\ldots,x_n)}$.

Base case ($d = 0$): A circuit of depth 0 computes either a constant function or a single input variable. For a constant $c \in \{0, 1\}$, the programs computing π^c, π^{-c}, and σ^c have length $O(1) = O(4^0)$. For an input variable x_i, we can construct programs of constant length that compute π^{x_i}, π^{-x_i}, and σ^{x_i} by testing x_i and applying the appropriate permutation.

Inductive step: Assume the statement holds for all circuits of depth at most $d - 1$. Consider a circuit of depth d computing $f(x_1, \ldots, x_n)$. The output gate must be either an AND or OR gate with inputs computing functions g and h, each computable by circuits of depth at most $d - 1$.

By the inductive hypothesis, there exist width 5 programs of length $O(4^{d-1})$ computing π^g, π^{-g}, σ^g, π^h, π^{-h}, and σ^h.

For the AND gate ($f = g \wedge h$), we use the identity:

$$\pi^{g\wedge h} = \pi^g\sigma^h\pi^{-g}\sigma^{-h}\pi^g\sigma^h\pi^{-g}\sigma^{-h}$$

This identity holds because when $g = h = 1$, the composition equals π (using $\pi\sigma\pi^{-1}\sigma^{-1} = \text{id}$ when π and σ commute in certain subgroups, and the double application), and equals the identity otherwise.

The length of this program is $8 \cdot O(4^{d-1}) = O(4^d)$. Similar constructions yield programs for π^{-f} and σ^f of length $O(4^d)$.

For the OR gate, we use De Morgan's laws and similar identities to construct the required programs.

Part 3. Given a Boolean circuit of depth d computing function f, we construct a branching program of width 5 and length $O(4^d)$ as follows.

By Part 2, there exists a width 5 program P of length $O(4^d)$ computing $\pi^{f(x_1,\ldots,x_n)}$. We modify this program to compute f itself.

Initialize the state to some fixed element $s_0 \in [5]$, say $s_0 = 1$. The program P computes a permutation that is either π (when $f = 1$) or the identity (when $f = 0$).

At the end of the program, we check whether the final state equals $\pi(s_0) = \pi(1)$ or $s_0 = 1$. If the final state is $\pi(1)$, output 1; if it is 1, output 0.

This branching program has width 5 (the state space is $[5]$) and length $O(4^d)$ (the length of program P), and correctly computes f. □

K.3.4 Gemini

Proof. **Part 1.** We consider the symmetric group S_5 acting on the set $[5] = \{1, 2, 3, 4, 5\}$. Let π and σ be the following 5-cycles:

$$\pi = (1\,2\,3\,4\,5), \quad \sigma = (1\,3\,5\,4\,2).$$

We compute the compositions $\pi\sigma$ and $\sigma\pi$:

$$\pi\sigma = (1\,2\,3\,4\,5)(1\,3\,5\,4\,2) = (1\,3\,2\,5\,4)$$

$$\sigma\pi = (1\,3\,5\,4\,2)(1\,2\,3\,4\,5) = (1\,4\,2\,3\,5)$$

Since $(1\,3\,2\,5\,4) \neq (1\,4\,2\,3\,5)$, we have $\pi\sigma \neq \sigma\pi$. Thus, these two permutations do not commute.

Part 2. We proceed by induction on the depth d of the Boolean circuit. We assume the circuit consists of binary AND and OR gates, and NOT gates acting on the inputs (literals).

Lemma (Commutators of 5-cycles): Since the alternating group A_5 is simple and non-abelian, for any 5-cycle $\gamma \in S_5$, there exist 5-cycles $\alpha, \beta \in S_5$ such that their commutator equals γ, i.e., $\alpha\beta\alpha^{-1}\beta^{-1} = \gamma$.

Base Case ($d = 0$): The function f is a literal, either x_i or $\neg x_i$. To compute ρ^f for any 5-cycle $\rho \in \{\pi, \pi^{-1}, \sigma\}$:

- If $f = x_i$, the program is a single instruction that applies ρ if $x_i = 1$ and the identity id if $x_i = 0$.
- If $f = \neg x_i$, the program is a single instruction that applies id if $x_i = 1$ and ρ if $x_i = 0$.

In both cases, the length is $1 = O(4^0)$.

Inductive Step: Assume that for any circuit of depth $d - 1$ computing a function g, and for any 5-cycle τ, there exists a width-5 program of length L_{d-1} computing τ^g. Consider a function f computed by a circuit of depth d.

Case 1: $f = g \wedge h$. We wish to compute $\gamma^f = \gamma^{g\wedge h}$ for a target 5-cycle γ. By the Lemma, there exist 5-cycles α, β such that $\alpha\beta\alpha^{-1}\beta^{-1} = \gamma$. By the inductive hypothesis, there exist programs $P_\alpha, P_\beta, P_{\alpha^{-1}}, P_{\beta^{-1}}$ computing $\alpha^g, \beta^h, \alpha^{-g}, \beta^{-h}$ respectively, each of length L_{d-1}. Construct the program P by concatenating these four programs:

$$P = P_\alpha P_\beta P_{\alpha^{-1}} P_{\beta^{-1}}.$$

The map computed by P is $\alpha^g\beta^h\alpha^{-g}\beta^{-h}$. We analyze the output based on the values of g and h:

- If $g = 0, h = 0$: $id \cdot id \cdot id \cdot id = id$.
- If $g = 0, h = 1$: $id \cdot \beta \cdot id \cdot \beta^{-1} = id$.
- If $g = 1, h = 0$: $\alpha \cdot id \cdot \alpha^{-1} \cdot id = id$.
- If $g = 1, h = 1$: $\alpha\beta\alpha^{-1}\beta^{-1} = \gamma$.

Thus, P computes $\gamma^{g\wedge h}$. The length is $4L_{d-1}$.

Case 2: $f = g \vee h$. By De Morgan's laws, $g \vee h = \neg(\neg g \wedge \neg h)$. We observe that for any function k and 5-cycle ρ, $\rho^{\neg k} = \rho \circ \rho^{-k}$. This can be implemented by appending a constant instruction applying ρ to the program for ρ^{-k}. To compute $\gamma^{g\vee h}$, we compute $\gamma \circ \gamma^{-(\neg g\wedge\neg h)}$. Let $\delta = \gamma^{-1}$. We need to compute $\delta^{\neg g\wedge\neg h}$. Using the construction from Case 1, we find commutators μ, ν for δ and sequence programs for $\mu^{\neg g}, \nu^{\neg h}, \mu^{-\neg g}, \nu^{-\neg h}$. Note that $\mu^{\neg g} = \mu \circ \mu^{-g}$. The total program involves 4 recursive calls for depth $d - 1$ (computing μ^{-g}, ν^{-h}, etc.) interspersed with constant permutations. The length satisfies $L_d \leq 4L_{d-1} + C$ for some constant C.

Solving the recurrence $L_d \leq 4L_{d-1} + C$ with $L_0 = 1$ yields $L_d = O(4^d)$. Thus, for any f of depth d, we can compute $\pi^f, \pi^{-f}, \sigma^f$ with width 5 and length $O(4^d)$.

Part 3. Given a Boolean circuit of depth d computing $f(x_1, \ldots, x_n)$, we apply the result from Part 2 to construct a width-5 program P computing π^f, where $\pi = (1\,2\,3\,4\,5)$. The length of P is $O(4^d)$.

We define the acceptance condition for the branching program as follows: The program starts in state $1 \in [5]$. For an input x:

- If $f(x) = 1$, the program computes the permutation π. The final state is $\pi(1) = 2$.
- If $f(x) = 0$, the program computes the identity permutation id. The final state is $id(1) = 1$.

Since $1 \neq 2$, the final state distinguishes between $f(x) = 0$ and $f(x) = 1$. Therefore, this branching program computes the function f. □

K.4 Problem 10.4

K.4.1 Problem Statement

Theorem K.4. *Consider the* perfect matching problem *for streaming algorithms. The input stream consists of edges in a graph with $2n$ vertices, and the algorithm needs to decide if the graph contains a matching of size n. Then $\Omega(n)$ space is required.*

K.4.2 GPT

Proof. We are tasked with proving that in the perfect matching problem for streaming algorithms, the space complexity is at least $\Omega(n)$, where the input stream consists of edges of a graph with $2n$ vertices, and the algorithm must decide if the graph contains a matching of size n.

Let us assume, for the sake of contradiction, that there exists a streaming algorithm that solves the perfect matching problem using $o(n)$ space. We will show that this leads to a contradiction.

Step 1: Space Requirement for Graph Representation The graph consists of $2n$ vertices, and the input stream contains edges. To detect a matching of size n, the algorithm must determine whether there exists a subset of n edges such that no two edges share a vertex. This requires the algorithm to maintain information about the relationships between vertices, including which vertices are matched, which vertices are unmatched, and potentially which edges have been seen.

Since we assume the algorithm uses $o(n)$ space, it can store only a limited amount of information. Specifically, it cannot store information for all $2n$ vertices or all possible edges that may appear in the stream. This severely limits the algorithm's ability to correctly determine whether a matching of size n exists.

Step 2: Counting Possible Matchings The total number of possible matchings in a graph with $2n$ vertices is given by the number of ways to choose n edges from the set of all edges, such that no two edges share a vertex. The number of such matchings is exponential in n, specifically on the order of $\binom{2n}{n}/n!$. This implies that the number of potential matchings grows rapidly as n increases.

Step 3: Information Theoretic Lower Bound In order to decide if a matching of size n exists, the algorithm must distinguish between a large number of possible configurations of edges. Each configuration corresponds to a distinct matching or a distinct way in which the vertices are connected by edges. Given that there are $\Omega(n!)$ possible matchings, the algorithm must be able to distinguish between these configurations.

By the information-theoretic argument, any algorithm that uses $o(n)$ space can store at most $o(n)$ bits of information. However, distinguishing between $\Omega(n!)$ distinct matchings requires at least $\log(\Omega(n!)) = \Omega(n \log n)$ bits of information. Therefore, the algorithm cannot distinguish between all possible configurations of edges using only $o(n)$ space.

Step 4: Conclusion Since the algorithm requires $\Omega(n \log n)$ bits of information to distinguish between all possible matchings, but it is restricted to $o(n)$ space, this leads to a contradiction. Thus, our initial assumption that there exists a streaming algorithm solving the perfect matching problem with $o(n)$ space is false.

Therefore, we conclude that any algorithm solving the perfect matching problem in the streaming model must use at least $\Omega(n)$ space.

□

K.4.3 Claude

Proof. We prove the lower bound by a reduction from the Index problem in communication complexity.

Let Alice hold a binary string $x \in \{0,1\}^m$ and Bob hold an index $i \in [m]$. In the Index problem, Bob must output x_i. It is well-known that any one-way communication protocol for Index requires $\Omega(m)$ bits of communication.

We construct a reduction as follows. Given an instance (x, i) of Index with $|x| = m$, we set $n = m + 1$ and construct a graph G with $2n = 2m + 2$ vertices as follows:

Define the vertex set as $V = \{a_0, a_1, \ldots, a_m\} \cup \{b_0, b_1, \ldots, b_m\}$.

Alice's edges: For each $j \in [m]$, if $x_j = 1$, Alice sends the edge (a_j, b_j) to the stream. Additionally, Alice sends the edge (a_0, b_0).

Bob's edges: Bob sends the edge (a_0, b_i) and the edge (a_i, b_0). For all $j \in [m] \setminus \{i\}$, Bob sends the edge (a_j, b_j).

We claim that G has a perfect matching if and only if $x_i = 1$.

Suppose $x_i = 1$. Then Alice sent the edge (a_i, b_i). We can construct a perfect matching as follows: match a_0 with b_0 using Alice's edge, match a_i with b_i using Alice's edge, and for each $j \in [m] \setminus \{i\}$, match a_j with b_j using Bob's edge. This gives a matching of size $m + 1 = n$.

Suppose $x_i = 0$. Then Alice did not send the edge (a_i, b_i). Consider any perfect matching M in G. The vertex a_0 must be matched to some vertex in $\{b_0, b_1, \ldots, b_m\}$. Similarly, b_0 must be matched to some vertex in $\{a_0, a_1, \ldots, a_m\}$.

If a_0 is matched to b_0, then a_i must be matched to some b_j and b_i must be matched to some a_k. The only edges incident to a_i are (a_i, b_0) from Bob (but b_0 is already matched) and potentially (a_i, b_i) from Alice (but this edge does not exist since $x_i = 0$). Thus a_i cannot be matched, which is a contradiction.

If a_0 is matched to b_i (using Bob's edge), then b_0 must be matched to a_i (using Bob's edge). But then for each $j \in [m] \setminus \{i\}$, we have a_j matched to b_j. However, a_i and b_i are already matched to b_0 and a_0 respectively, so all vertices are matched. But this requires the edges (a_0, b_i), (a_i, b_0), and (a_j, b_j) for all $j \neq i$, which exist. Yet we need to verify: a_i is matched to b_0, so b_i must be matched to some vertex. The only available vertex is a_0, which is matched to b_i. This accounts for all vertices and forms a valid matching.

We reconsider: if $x_i = 0$, the edge (a_i, b_i) is missing. The edges from Bob include (a_0, b_i), (a_i, b_0), and (a_j, b_j) for $j \neq i$. Combined with Alice's edge (a_0, b_0) and (a_j, b_j) for j where $x_j = 1$, we see that vertices a_i and b_i cannot both be matched unless the edge (a_i, b_i) exists. Therefore, no perfect matching exists when $x_i = 0$.

Thus, a streaming algorithm that solves perfect matching with space s gives a one-way communication protocol for Index with communication s. Since Index requires $\Omega(m)$ bits, we have $s = \Omega(m) = \Omega(n)$. □

K.4.4 Gemini

Proof. We establish the lower bound via a reduction from the one-way communication complexity of the INDEX problem. In the INDEX problem, Alice possesses a string $x \in \{0,1\}^n$, and Bob possesses an index $k \in \{1, \ldots, n\}$. Alice sends a single message to Bob, who must determine the value of x_k. It is a standard result in communication complexity that any randomized protocol that solves INDEX with constant error probability requires $\Omega(n)$ bits of communication.

Suppose there exists a streaming algorithm $\mathcal{A}$ that decides the perfect matching problem on a graph with $2n$ vertices using $S(n)$ bits of space. We construct a one-way communication protocol for INDEX using $\mathcal{A}$.

Alice and Bob agree on a fixed set of $2n$ vertices, labeled $V = \{u_1, \ldots, u_n, v_1, \ldots, v_n\}$. Given input x, Alice initializes $\mathcal{A}$ and processes a stream of edges determined by x. For each $i \in \{1, \ldots, n\}$, if $x_i = 1$, Alice inserts the edge (u_i, v_i) into the stream. If $x_i = 0$, she adds no edge for index i. After processing these edges, Alice sends the memory state of $\mathcal{A}$ to Bob. The size of this transmission is $S(n)$ bits.

Bob receives the memory state and continues the execution of $\mathcal{A}$. Based on his input k, Bob inserts an edge (u_j, v_j) into the stream for every $j \in \{1, \ldots, n\}$ such that $j \neq k$. Bob then queries $\mathcal{A}$ to determine if the resulting graph $G = (V, E)$ contains a perfect matching and outputs the algorithm's decision.

We analyze the graph G constructed by this protocol. The edge set E is the union of edges added by Alice and Bob. Specifically, the potential edges are restricted to the disjoint pairs $\{(u_i, v_i)\}_{i=1}^n$. A graph

consisting of disjoint pairs has a perfect matching if and only if every pair is connected by an edge. For any $j \neq k$, the edge (u_j, v_j) is added by Bob, ensuring that the pair $\{u_j, v_j\}$ is matched. For the specific index k, Bob does not add an edge. Thus, the edge (u_k, v_k) exists in G if and only if Alice added it, which occurs if and only if $x_k = 1$.

Therefore, G contains a perfect matching if and only if $(u_k, v_k) \in E$, which is equivalent to $x_k = 1$. Since $\mathcal{A}$ decides the perfect matching problem correctly, Bob correctly outputs x_k. The space complexity $S(n)$ of the streaming algorithm corresponds to the communication cost of the protocol. By the $\Omega(n)$ lower bound for INDEX, we conclude that $S(n) = \Omega(n)$. □

L Problems in Chapter 11

L.1 Problem 11.1

L.1.1 Problem Statement

Theorem L.1. *Modify the van Emde Boas tree data structure so that it can maintain the median of n numbers, with time $O(\log \log n)$ for adding, deleting, and querying the median.*

L.1.2 GPT

Proof. We are tasked with modifying the van Emde Boas (vEB) tree data structure to maintain the median of a set of n numbers, while ensuring that the time complexity for adding, deleting, and querying the median remains $O(\log \log n)$. We will achieve this by enhancing the structure of the vEB tree with additional operations that allow efficient median maintenance.

Let T be a vEB tree storing n elements, and let M be the median of the elements currently in the tree. The key idea is to maintain two balanced subtrees that divide the elements into two groups: those smaller than M and those greater than M. Additionally, we will maintain a pointer to the current median, allowing us to efficiently retrieve and update the median during insertions and deletions.

Step 1: Modifying the vEB tree structure

The original vEB tree is defined for a universe of size u, with u being a power of 2. The tree recursively splits into smaller trees, with the top-level tree (of size $u/2$) storing the higher-order bits and the lower-level tree (of size $u/2$) storing the lower-order bits. We modify this structure to maintain the median by splitting the tree into two parts: 1. A subtree T_{low} that stores all elements less than M. 2. A subtree T_{high} that stores all elements greater than M.

Let n_{low} and n_{high} represent the number of elements in the lower and higher subtrees, respectively. The tree is balanced in such a way that $n_{\text{low}} \approx n_{\text{high}}$.

Step 2: Insertion and Deletion

Insertion:

When inserting an element x into the tree, we first compare x with the current median M. - If $x < M$, we insert it into the subtree T_{low}. - If $x > M$, we insert it into the subtree T_{high}. - If $x = M$, we insert it directly at the root.

After the insertion, we need to adjust the median. If the size of T_{low} has increased, and the number of elements in T_{low} exceeds half of the total size of the tree, then we move the largest element from T_{low} to M and adjust the subtree sizes accordingly. Similarly, if T_{high} has increased, we move the smallest element from T_{high} to M.

Deletion:

When deleting an element x from the tree: - If $x < M$, we delete it from T_{low}. - If $x > M$, we delete it from T_{high}. - If $x = M$, we remove M from the tree and reassign a new median, which will be either the largest element from T_{low} or the smallest element from T_{high}, depending on the tree's balance.

Step 3: Querying the Median

To query the median, we simply return the value stored in M. This operation takes constant time, as we always have a direct reference to the median.

Step 4: Time Complexity Analysis

- **Insertion and Deletion**: Both insertion and deletion involve traversing the tree structure, which requires $O(\log \log n)$ time in a vEB tree. Since inserting or deleting an element in either the lower or

upper subtree also takes $O(\log \log n)$ time, and the median adjustment involves only a constant amount of work, the overall time complexity remains $O(\log \log n)$. - **Querying the Median**: Since the median is directly accessible at all times, querying the median takes constant time, i.e., $O(1)$.

Thus, the modified vEB tree supports insertion, deletion, and median query operations in $O(\log \log n)$ time.

□

L.1.3 Claude

Proof. We modify the van Emde Boas tree data structure to maintain the median of n numbers with $O(\log \log n)$ time complexity for insertion, deletion, and median queries.

Let U denote the universe size. A van Emde Boas tree supports insertion, deletion, and successor/predecessor queries in $O(\log \log U)$ time. We augment this structure to track the median efficiently.

Data Structure Augmentation:

We maintain two van Emde Boas trees, denoted T_{low} and T_{high}, along with a counter n for the total number of elements. The invariant is that T_{low} contains the smaller half of the elements and T_{high} contains the larger half, with the property that $|T_{\text{low}}| - |T_{\text{high}}| \in \{0, 1\}$ and every element in T_{low} is at most every element in T_{high}.

Additionally, we store n_{low} and n_{high} as the sizes of T_{low} and T_{high} respectively, and maintain that $n = n_{\text{low}} + n_{\text{high}}$.

The median is defined as the maximum element in T_{low} when n is odd (where $n_{\text{low}} = n_{\text{high}} + 1$), or the maximum of T_{low} when n is even (where $n_{\text{low}} = n_{\text{high}}$).

Query Operation:

To find the median, we return $\max(T_{\text{low}})$, which can be computed in $O(\log \log U)$ time using the van Emde Boas tree maximum operation. Since $U \leq 2^n$ for n elements, we have $\log \log U \leq \log \log 2^n = \log n$, but for a fixed universe size or when $U = O(n^c)$ for constant c, this is $O(\log \log n)$.

Insertion Operation:

To insert element x:

(1) If $n = 0$, insert x into T_{low}, set $n_{\text{low}} = 1$, $n_{\text{high}} = 0$, and $n = 1$. Time: $O(\log \log U)$.

(2) If $n > 0$, let $m = \max(T_{\text{low}})$ be the current median element.

(3) If $x \leq m$, insert x into T_{low}. Otherwise, insert x into T_{high}. Time: $O(\log \log U)$.

(4) Update n_{low} or n_{high} accordingly and set $n = n_{\text{low}} + n_{\text{high}}$.

(5) Rebalance if necessary: If $n_{\text{low}} > n_{\text{high}} + 1$, move $m' = \max(T_{\text{low}})$ from T_{low} to T_{high} by deleting from T_{low} and inserting into T_{high}. If $n_{\text{high}} > n_{\text{low}}$, move $m'' = \min(T_{\text{high}})$ from T_{high} to T_{low}. Each rebalancing operation takes $O(\log \log U)$ time.

Total time for insertion: $O(\log \log U) = O(\log \log n)$ when $U = \text{poly}(n)$.

Deletion Operation:

To delete element x:

(1) Determine whether $x \in T_{\text{low}}$ or $x \in T_{\text{high}}$ by comparing with $\max(T_{\text{low}})$.

(2) Delete x from the appropriate tree in $O(\log \log U)$ time.

(3) Update counters n_{low}, n_{high}, and n.

(4) Rebalance using the same procedure as in insertion to maintain the invariant. Time: $O(\log \log U)$.

Total time for deletion: $O(\log \log U) = O(\log \log n)$.

Correctness:

The invariant ensures that the maximum of T_{low} is always the median (or one of the two middle elements). The rebalancing operations maintain $|n_{\text{low}} - n_{\text{high}}| \leq 1$, which guarantees that $\max(T_{\text{low}})$ correctly represents the median.

Therefore, the modified van Emde Boas tree maintains the median with $O(\log \log n)$ time complexity for all three operations. □

L.1.4 Gemini

Proof. Let U be the size of the universe of numbers. We assume the standard computational model for van Emde Boas (vEB) trees, where the basic operations run in $O(\log \log U)$ time. While the theorem

statement refers to n numbers, the time complexity of vEB structures is dependent on the universe size U. We construct a modified data structure that satisfies the requirements with time complexity $O(\log \log U)$.

Data Structure Construction: We define the modified data structure $\mathcal{D}$ to consist of two standard van Emde Boas trees, denoted as T_{low} and T_{high}. Additionally, we maintain two integer counters, N_{low} and N_{high}, representing the number of elements stored in T_{low} and T_{high} respectively.

Invariants: Let S be the multiset of n numbers currently stored in $\mathcal{D}$. We maintain the following invariants: 1. $S = T_{low} \cup T_{high}$ (viewed as multisets). 2. For every $x \in T_{low}$ and $y \in T_{high}$, $x \leq y$. 3. The sizes are balanced such that $N_{low} = \lceil n/2 \rceil$ and $N_{high} = \lfloor n/2 \rfloor$.

Under these invariants, the median of S corresponds to the maximum element stored in T_{low}.

Operations:

1. **Query Median:** The median is given by $\max(T_{low})$. Standard vEB trees maintain the maximum element (often denoted as *max*) as a stored attribute.

$$\text{Time} = O(1).$$

2. **Insert(x):** To insert an element x:

- If T_{low} is empty, insert x into T_{low} and increment N_{low}.
- Otherwise, let $m = \max(T_{low})$. If $x \leq m$, insert x into T_{low} and increment N_{low}. If $x > m$, insert x into T_{high} and increment N_{high}.

Rebalancing: After insertion, the size invariant may be violated by at most 1.

- If $N_{low} > \lceil n/2 \rceil$ (i.e., $N_{low} = N_{high} + 2$ if n is even, or similar imbalance), we move the largest element of T_{low} to T_{high}.
  1. $val \leftarrow \max(T_{low})$.
  2. Delete(T_{low}, val).
  3. Insert(T_{high}, val).
  4. Decrement N_{low}, increment N_{high}.
- If $N_{low} < \lceil n/2 \rceil$ (i.e., $N_{high} > N_{low}$), we move the smallest element of T_{high} to T_{low}.
  1. $val \leftarrow \min(T_{high})$.
  2. Delete(T_{high}, val).
  3. Insert(T_{low}, val).
  4. Decrement N_{high}, increment N_{low}.

Since standard vEB ‘Insert‘, ‘Delete‘, ‘Min‘, and ‘Max‘ operations take $O(\log \log U)$ time, and we perform a constant number of these operations, the total time is $O(\log \log U)$.

3. **Delete(x):** To delete an element x:

- Compare x with $m = \max(T_{low})$.
- If $x \leq m$, delete x from T_{low} and decrement N_{low}.
- If $x > m$, delete x from T_{high} and decrement N_{high}.

Rebalancing: Similar to insertion, check the invariants.

- If $N_{low} < \lceil n/2 \rceil$, move $\min(T_{high})$ to T_{low}.
- If $N_{low} > \lceil n/2 \rceil$, move $\max(T_{low})$ to T_{high}.

This requires a constant number of standard vEB operations.

$$\text{Time} = O(\log \log U).$$

Conclusion: The data structure $\mathcal{D}$ maintains the median of the set S at all times. The time complexity for 'Insert', 'Delete', and 'Query Median' is dominated by the standard van Emde Boas operations, which is $O(\log \log U)$. Assuming the standard context where the universe size U is polynomially related to n (or simply that the problem statement implies the vEB complexity class), the operations run in the required asymptotic time. □

L.2 Problem 11.2

L.2.1 Problem Statement

Theorem L.2. *We showed that every static data structure problem leads to a communication game and used lower bounds on the total number of bits communicated in the game to prove data structure lower bounds. Show that if the number of possible queries to the data structure is Q, then we will never be able to prove that $s > (2Q)^{1/t}/2^w$ using such an approach. Conclude that any lower bound showing that $t \gg \log Q$ must use a different approach.*

L.2.2 GPT

Proof. Let the number of possible queries to the data structure be denoted by Q. The static data structure problem can be modeled as a communication game, where the number of bits communicated during the game provides a lower bound for the query complexity of the problem. The objective of the proof is to show that it is impossible to prove the inequality $s > \frac{(2Q)^{1/t}}{2^w}$ using the communication game approach.

First, recall that the communication game model for a static data structure problem involves the communication of bits between the players in the game, which is bounded by the total number of bits exchanged. The number of bits communicated during the game is related to the size of the state of the data structure, which is denoted by s. The total number of possible queries, Q, places an upper bound on the amount of information that can be extracted by any query.

Next, consider the expression $\frac{(2Q)^{1/t}}{2^w}$. In order to prove that $s > \frac{(2Q)^{1/t}}{2^w}$, we need to establish a relationship between s, Q, t, and w in the context of the communication game. However, the communication game lower bounds the number of bits that must be exchanged, and the number of bits exchanged depends on Q and s. Specifically, the total communication is bounded by a function of Q and s, and we cannot prove that s exceeds the expression $\frac{(2Q)^{1/t}}{2^w}$ under this model because of the following constraint on the number of bits communicated.

The number of bits communicated in the game is upper-bounded by a function of the form $O(\log Q)$, where the logarithm reflects the number of bits needed to represent the possible queries. On the other hand, the expression $\frac{(2Q)^{1/t}}{2^w}$ grows at a different rate than the logarithmic growth of the communication bits. In particular, the rate of growth of this expression with respect to Q is not sufficient to allow the conclusion that s can exceed it using the communication game framework.

Therefore, it is impossible to prove that $s > \frac{(2Q)^{1/t}}{2^w}$ using only the communication game approach. This implies that any lower bound showing that $t \gg \log Q$ must necessarily use a different approach, as the communication game model cannot provide such a lower bound due to the limitations of the communication and the growth rates involved.

Thus, the proof is complete. □

L.2.3 Claude

Proof. We prove the theorem in two parts: first establishing the upper bound on s, then deriving the consequence for lower bounds with $t \gg \log Q$.

Part 1: Upper bound on s via communication complexity.

Consider a static data structure problem with Q possible queries. In the communication game framework, Alice receives the data structure content and Bob receives a query. The goal is for Bob to learn the answer to his query.

We establish a trivial communication protocol that provides an upper bound on the communication complexity. Alice can simply send Bob the entire data structure, which requires at most $s \cdot w$ bits of communication, where s is the space (number of words) and w is the word size. Bob can then answer his query locally in time t.

Alternatively, consider a protocol where Bob sends his query to Alice (requiring $\lceil \log Q \rceil$ bits since there are Q possible queries), and Alice responds with the answer. The answer requires at most w bits. This gives a total communication of at most $\log Q + w$ bits in a single round.

For a t-round protocol, we can achieve communication complexity C satisfying:

$$C \leq \min(s \cdot w, t \cdot w + \log Q)$$

More generally, in any t-round protocol, the total communication is bounded by the minimum of sending the entire data structure or conducting t rounds of interaction. A balanced protocol might use $s \cdot w/t$ bits per round.

Now, suppose we have a communication lower bound showing that any protocol requires at least C bits of total communication. From the trivial upper bound protocol, we know:

$$C \leq s \cdot w$$

Furthermore, there exists a trivial protocol using t rounds where in each round we can communicate $O(w)$ bits, and Bob can specify which part of the data structure he needs. Since there are Q possible queries, Bob needs to identify his query (requiring $\log Q$ bits) and receive an answer (requiring w bits). Over t rounds, we can achieve:

$$C \leq t \cdot w + \log Q$$

For the communication game to yield a meaningful lower bound on s, we need C to be large. However, the trivial protocol gives us:

$$C \leq \min(s \cdot w, t \cdot w)$$

Since we can always answer any query by examining at most the entire data structure, and there are Q possible queries, information theory gives us that the communication complexity satisfies:

$$C \leq \log(2^{sw}) = s \cdot w$$

Moreover, in t rounds, each round can communicate at most $s \cdot w/t$ bits on average. The total information that can distinguish between Q queries is bounded by:

$$C \leq t \cdot 2^w$$

since in each round, at most 2^w different messages (of size w bits) can be sent.

Combining these constraints, any communication lower bound C must satisfy:

$$C \leq \min(s \cdot w, t \cdot 2^w)$$

To prove $s > s_0$ for some threshold s_0, we need $C > s_0 \cdot w$. But we also have $C \leq t \cdot 2^w$. Therefore:

$$s_0 \cdot w < C \leq t \cdot 2^w$$

This implies:

$$s_0 < \frac{t \cdot 2^w}{w}$$

Additionally, since there are only Q possible queries, the communication complexity is bounded by $C \leq \log Q + w$ (Bob identifies his query and receives an answer). In a t-round protocol, we have:

$$C \leq t \cdot 2^w$$

and also $C \leq s \cdot w$ from the trivial protocol. For the lower bound to be non-trivial, we need:

$$s \cdot w > t \cdot 2^w$$

which gives:

$$s > \frac{t \cdot 2^w}{w}$$

However, the fundamental limitation comes from the fact that with Q queries, we can use a lookup table of size $O(Q)$ to answer any query in constant time. This means:

$$s \leq O(Q)$$

The communication complexity approach cannot distinguish between data structures of size larger than $(2Q)^{1/t}/2^w$ because the trivial protocol using t rounds can achieve communication $t \cdot 2^w$, and this must be at least $s \cdot w$ for a meaningful lower bound. Setting $s \cdot w = t \cdot 2^w$ gives $s = t \cdot 2^w/w$. But we also need $s \leq Q$, so:

$$s \leq \min\left(Q, \frac{t \cdot 2^w}{w}\right)$$

Therefore, we cannot prove $s > (2Q)^{1/t}/2^w$ using this approach.

Part 2: Consequence for $t \gg \log Q$.

Suppose we want to prove a lower bound showing that $t \gg \log Q$. From Part 1, the communication complexity approach can only prove:

$$s \leq \frac{(2Q)^{1/t}}{2^w}$$

If $t \gg \log Q$, then $t = \omega(\log Q)$, which means $(2Q)^{1/t} = 2^{O(\log Q/t)} = 2^{o(1)} = 1 + o(1)$. Therefore:

$$s \leq \frac{1 + o(1)}{2^w} = o(1)$$

This bound is trivial and provides no meaningful information about the data structure space requirement. Therefore, any lower bound showing that $t \gg \log Q$ must use a different approach than communication complexity of this form. □

L.2.4 Gemini

Proof. Let the static data structure problem be modeled as a communication game between two parties: Alice, who holds the query $q \in \{1, \ldots, Q\}$, and Bob, who holds the data structure D. The data structure consists of s cells (words), each of size w bits.

In the cell-probe model, an algorithm answering a query performs t probes to the memory. This process can be simulated by a communication protocol where, for each probe: 1. Alice sends the address of the cell to Bob. Since there are s cells, this requires $\log s$ bits. 2. Bob sends the content of the cell to Alice. This requires w bits.

Thus, the total number of bits communicated in this simulation protocol, denoted C_{probe}, is:

$$C_{\text{probe}} = t(\log s + w).$$

However, there exists a trivial protocol for any such problem. Alice can simply transmit the query index q to Bob. Bob, having access to the entire data structure, computes the answer and sends it back to Alice. The query q requires $\lceil \log Q \rceil$ bits. Assuming the answer is a single bit (or a small constant absorbed into the bound), the total communication cost of this trivial protocol, C_{triv}, is bounded by:

$$C_{\text{triv}} \leq \lceil \log Q \rceil + 1.$$

Using the property that $x + 1 = \log(2^x) + \log 2 = \log(2 \cdot 2^x)$, we write the upper bound as:

$$C_{\text{triv}} \leq \log(2Q).$$

The communication complexity approach establishes data structure lower bounds by proving that the problem requires a minimum amount of communication, say L bits. Specifically, one shows that any valid protocol must satisfy $C \geq L$. Since the probe simulation is a valid protocol, we deduce $C_{\text{probe}} \geq L$.

However, the true communication complexity of the problem is bounded from above by the most efficient protocol available. Therefore, any provable lower bound L on the communication complexity must satisfy $L \leq C_{\text{triv}}$. It follows that this approach can only establish bounds consistent with the inequality:

$$C_{\text{probe}} \leq C_{\text{triv}}.$$

Substituting the expressions for the costs:

$$t(\log s + w) \leq \log(2Q).$$

We rearrange this inequality to solve for the space s:

$$\begin{aligned}
\log s + w &\leq \frac{\log(2Q)}{t} \\
\log s &\leq \log\left((2Q)^{1/t}\right) - w \\
\log s &\leq \log\left((2Q)^{1/t}\right) - \log(2^w) \\
\log s &\leq \log\left(\frac{(2Q)^{1/t}}{2^w}\right).
\end{aligned}$$

Exponentiating both sides, we obtain:

$$s \leq \frac{(2Q)^{1/t}}{2^w}.$$

Since the method relies on the communication cost of the probes not exceeding the trivial communication upper bound, we can never prove a statement that contradicts this inequality. Thus, we cannot prove $s > (2Q)^{1/t}/2^w$ using this approach.

Finally, consider the implication for the time complexity t. If we attempt to prove a lower bound where $t \gg \log Q$, then as $Q \to \infty$, the term $1/t$ becomes negligible. Consequently, $(2Q)^{1/t} \to 1$. The inequality then approaches $s \leq 2^{-w}$. For any practical word size $w \geq 1$, this implies $s < 1$, which is impossible for a non-empty data structure. Therefore, any lower bound showing $t \gg \log Q$ is unattainable via this communication game reduction and must use a different approach. □

L.3 Problem 11.3

L.3.1 Problem Statement

Theorem L.3. *For $\delta > 0$, n operations of a dynamic data structure with parameters s, t_u, t_q, w and error $\epsilon > 0$ can always be simulated by another data structure with space $n^2(t_u + t_q)/\delta$, update time t_u, query time t_q, word size w, and error $\epsilon + \delta$.*

L.3.2 GPT

Proof. Let the original dynamic data structure be denoted by $\mathcal{D}$ with parameters s, t_u, t_q, w, and error $\epsilon > 0$. We wish to show that this data structure can be simulated by another data structure with space $n^2(t_u + t_q)/\delta$, update time t_u, query time t_q, word size w, and error $\epsilon + \delta$ for some $\delta > 0$.

First, consider the original data structure $\mathcal{D}$, which performs n operations, where each operation consists of an update and a query. Let the update time for $\mathcal{D}$ be denoted by t_u, and the query time by t_q. The space complexity of $\mathcal{D}$ is s, and the error associated with the data structure is bounded by $\epsilon > 0$.

To simulate $\mathcal{D}$ with the new data structure, we introduce a new data structure $\mathcal{D}'$ with the following parameters: - Space complexity: $n^2(t_u + t_q)/\delta$, - Update time: t_u, - Query time: t_q, - Word size: w, - Error: $\epsilon + \delta$.

We will now describe how the simulation works and verify the correctness of the parameters.

Space Complexity The space complexity of the new data structure $\mathcal{D}'$ is $n^2(t_u + t_q)/\delta$. To understand this, note that the original data structure $\mathcal{D}$ operates with space s, which is a function of n, the number of operations. The space complexity of $\mathcal{D}'$ is scaled by a factor of $n^2(t_u + t_q)/\delta$ to account for the increase in error tolerance from ϵ to $\epsilon + \delta$.

Update and Query Times The update time of $\mathcal{D}'$ is the same as the update time of $\mathcal{D}$, i.e., t_u, and similarly, the query time of $\mathcal{D}'$ is the same as that of $\mathcal{D}$, i.e., t_q. These times are preserved because the underlying operations of $\mathcal{D}$ are directly mimicked by $\mathcal{D}'$, with no additional overhead introduced by the simulation process.

Error Bound The error bound of the new data structure $\mathcal{D}'$ is increased from ϵ to $\epsilon + \delta$. This increase in error is a consequence of the additional δ introduced in the simulation process. The extra error does not significantly affect the correctness of the simulated operations, but it ensures that the error in $\mathcal{D}'$ is within acceptable bounds.

Conclusion Thus, the new data structure $\mathcal{D}'$ simulates the original data structure $\mathcal{D}$ with the desired parameters: space complexity of $n^2(t_u + t_q)/\delta$, update time t_u, query time t_q, word size w, and error $\epsilon + \delta$. Therefore, the theorem holds.

□

L.3.3 Claude

Proof. We construct a simulation that maintains multiple independent copies of the original data structure to reduce the error probability through redundancy and majority voting.

Let D be the original dynamic data structure with space s, update time t_u, query time t_q, word size w, and error probability ϵ for each operation. We construct a new data structure D' as follows.

Define $k = \lceil n(t_u + t_q)/\delta \rceil$ to be the number of independent copies of D that we maintain. Since each operation (update or query) on the original structure takes time at most $t_u + t_q$, and we perform n operations total, the total number of individual operations across all copies is bounded by $n \cdot k$.

For each update operation on D', we perform the same update on all k copies independently. This requires time $k \cdot t_u = O(t_u)$ per copy, giving total update time t_u when amortized appropriately.

For each query operation on D', we query all k copies independently and return the majority answer (or an arbitrary answer if no majority exists). This requires time $k \cdot t_q = O(t_q)$ per copy, giving total query time t_q when amortized appropriately.

The space requirement is $k \cdot s$ for storing all copies. Since $k = \lceil n(t_u + t_q)/\delta \rceil \leq n(t_u + t_q)/\delta + 1$, and assuming $s \leq n$ (as the structure must store information about n operations), we have total space at most $(n(t_u + t_q)/\delta + 1) \cdot n = n^2(t_u + t_q)/\delta + n \leq 2n^2(t_u + t_q)/\delta$ for sufficiently large n.

Now we analyze the error probability. Consider any sequence of n operations. Each individual operation on a single copy has error probability at most ϵ. By the union bound, the probability that any specific copy makes an error in any of the n operations is at most $n\epsilon$.

For a query operation, let X_i be the indicator that copy i returns an incorrect answer. Then $\mathbb{E}[X_i] \leq \epsilon$ for each query. By independence of the copies and Markov's inequality, the probability that more than half of the k copies return incorrect answers is bounded by:

$$\Pr\left[\sum_{i=1}^{k} X_i > k/2\right] \leq \frac{2\mathbb{E}[\sum_{i=1}^{k} X_i]}{k} \leq \frac{2k\epsilon}{k} = 2\epsilon$$

However, we need a tighter analysis. By Chernoff bounds, for independent copies with error probability ϵ each, the probability that the majority vote fails is exponentially small in k when $\epsilon < 1/2$.

More precisely, using the union bound over all n operations and the fact that $k = \lceil n(t_u + t_q)/\delta \rceil$, the total error probability across all n operations is at most:

$$n \cdot \exp(-\Omega(k)) + n\epsilon \leq \delta + n\epsilon$$

By choosing the constant in k appropriately, we ensure the first term is at most δ, giving total error probability at most $\epsilon + \delta$.

Therefore, D' simulates n operations with space $n^2(t_u + t_q)/\delta$, update time t_u, query time t_q, word size w, and error probability $\epsilon + \delta$. □

L.3.4 Gemini

Proof. Let $\mathcal{D}$ be the original dynamic data structure with space parameter s, update time t_u, query time t_q, word size w, and error probability ϵ. We construct a simulating data structure $\mathcal{D}'$ as follows.

Simulation Strategy: Instead of maintaining the full memory space of size s (which may be very large), $\mathcal{D}'$ maintains a hash table H of size S to store the contents of the memory cells accessed by $\mathcal{D}$. We use a universal hash family $\mathcal{H}$ mapping the address space $\{0, \ldots, s-1\}$ to the table indices $\{0, \ldots, S-1\}$. We select a hash function $h \in \mathcal{H}$ uniformly at random at initialization.

For every write operation to address x with value v in $\mathcal{D}$, $\mathcal{D}'$ writes v to $H[h(x)]$. For every read operation from address x in $\mathcal{D}$, $\mathcal{D}'$ reads the value from $H[h(x)]$. If the cell $H[h(x)]$ is empty, it returns the default initial value (e.g., 0).

Parameters of $\mathcal{D}'$:

1. **Space:** We set the size of the hash table to be $S = \lceil n^2(t_u + t_q)/\delta \rceil$ words. This matches the space requirement in the theorem statement.

2. **Time:** Assuming the hash function h can be evaluated in $O(1)$ time and table accesses take $O(1)$ time, the update and query times for $\mathcal{D}'$ remain t_u and t_q respectively.

3. **Word Size:** The word size w is preserved as we store the same values.

Error Analysis: The simulation $\mathcal{D}'$ behaves exactly like $\mathcal{D}$ unless a collision occurs in the hash table. A collision occurs if two distinct addresses x, y accessed during the sequence of n operations map to the same index, i.e., $h(x) = h(y)$.

Let $\mathcal{A}$ be the set of all distinct memory addresses accessed by $\mathcal{D}$ during the sequence of n operations. Since each update takes time t_u and each query takes time t_q, the total number of memory accesses (and thus the size of $\mathcal{A}$) is bounded by:

$$|\mathcal{A}| \leq n(t_u + t_q).$$

Let $N = n(t_u + t_q)$. The simulation fails due to hashing if there is a collision among the addresses in $\mathcal{A}$. Using a universal hash function, the probability that any specific pair of distinct keys collides is at most $1/S$. By the union bound over all pairs in $\mathcal{A}$, the probability of a collision, denoted $\Pr[\text{Collision}]$, is:

$$\Pr[\text{Collision}] \leq \binom{|\mathcal{A}|}{2} \frac{1}{S} < \frac{|\mathcal{A}|^2}{2S} \leq \frac{N^2}{2S}.$$

Substituting $N = n(t_u + t_q)$ and the allocated space $S = n^2(t_u + t_q)/\delta$:

$$\Pr[\text{Collision}] < \frac{(n(t_u + t_q))^2}{2 \cdot \frac{n^2(t_u+t_q)}{\delta}} = \frac{n^2(t_u + t_q)^2 \cdot \delta}{2n^2(t_u + t_q)} = \frac{\delta(t_u + t_q)}{2}.$$

Assuming the standard normalization where the operational parameters $t_u, t_q \geq 1$ are constants relative to n, or taking the conservative bound for the simulation lemma where the space is sufficient to suppress collisions, the probability of failure introduced by the simulation is bounded by δ.

The total error probability of $\mathcal{D}'$ is the probability that $\mathcal{D}$ fails (which is ϵ) plus the probability that the simulation fails (which is at most δ). Thus, the total error is at most $\epsilon + \delta$.

We have constructed a data structure $\mathcal{D}'$ with space $n^2(t_u + t_q)/\delta$, update time t_u, query time t_q, word size w, and error $\epsilon + \delta$, which simulates n operations of $\mathcal{D}$. □

L.4 Problem 11.4

L.4.1 Problem Statement

Theorem L.4. *For the k-lopsided set intersection data structure problem, one cannot improve on the trivial solution for $k = 1$.*

L.4.2 GPT

Proof. Fix a finite universe U with $|U| = m \geq 1$. In the k-lopsided set intersection data structure problem, one is given, in a preprocessing stage, a set $S \subseteq U$ (the stored set), and in a query stage one is given a query set $Q \subseteq U$ with $|Q| \leq k$; the required output is the intersection $S \cap Q$.

Assume $k = 1$. Then every query set has the form $Q = \{x\}$ for some $x \in U$, and the required output is

$$S \cap \{x\} = \begin{cases} \{x\} & \text{if } x \in S, \\ \emptyset & \text{if } x \notin S. \end{cases}$$

Thus, solving the 1-lopsided set intersection problem is equivalent to solving the static membership problem for S: given $x \in U$, decide whether $x \in S$, and output $\{x\}$ or $\emptyset$ accordingly.

Consider any (deterministic) data structure for this problem. Formally, let $\mathcal{P}$ be a preprocessing algorithm which, given $S \subseteq U$, produces a memory representation $M(S)$, and let $\mathcal{Q}$ be a query algorithm which, given $x \in U$ and oracle access to $M(S)$, outputs a set $A(x, S) \subseteq U$ that is required to equal $S \cap \{x\}$ for all $S \subseteq U$ and all $x \in U$. Let the query time be measured by the number of probes made by $\mathcal{Q}$ to the memory representation $M(S)$, in the standard cell-probe sense (the argument below applies a fortiori to any stronger model of computation, since having fewer probes is more restrictive).

Suppose, for contradiction, that $\mathcal{Q}$ makes 0 probes on every input. Then, for each $x \in U$, the output of $\mathcal{Q}$ is independent of the stored set S, because $\mathcal{Q}$ receives no information from $M(S)$. Hence there exists a function $f : U \to \{\emptyset, \{x\}\}$ such that for every $S \subseteq U$ and every $x \in U$ one has $A(x, S) = f(x)$.

Fix any $x_0 \in U$. If $f(x_0) = \emptyset$, then taking $S = \{x_0\}$ yields the correctness requirement $A(x_0, S) = S \cap \{x_0\} = \{x_0\}$, contradicting $A(x_0, S) = f(x_0) = \emptyset$. If instead $f(x_0) = \{x_0\}$, then taking $S = \emptyset$ yields the requirement $A(x_0, S) = S \cap \{x_0\} = \emptyset$, contradicting $A(x_0, S) = f(x_0) = \{x_0\}$. In either case a contradiction is obtained. Therefore $\mathcal{Q}$ cannot make 0 probes on all inputs, and so every correct data structure for the 1-lopsided set intersection problem must have worst-case query time at least 1 probe.

On the other hand, there is a trivial solution with worst-case query time 1: store the characteristic function $\chi_S : U \to \{0, 1\}$ in memory, and on query $\{x\}$ probe the single bit $\chi_S(x)$; output $\{x\}$ if $\chi_S(x) = 1$ and $\emptyset$ otherwise. This achieves one probe per query.

Consequently, for $k = 1$ the worst-case query time of the trivial solution is optimal, and thus one cannot improve on the trivial solution for $k = 1$ in the k-lopsided set intersection data structure problem. □

L.4.3 Claude

Proof. We prove that for $k = 1$, the trivial solution for the lopsided set intersection data structure problem is optimal.

Recall that in the k-lopsided set intersection problem, we preprocess a collection of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ where each $S_i \subseteq U$ for some universe U, and we must answer queries of the form: given a query set Q with $|Q| \leq k$, report all sets $S_i \in \mathcal{S}$ such that $S_i \cap Q \neq \emptyset$.

For $k = 1$, each query set Q contains exactly one element, say $Q = \{q\}$ for some $q \in U$. The query asks for all sets S_i that contain the element q.

The trivial solution maintains an inverted index: for each element $u \in U$, store a list $L(u)$ of all sets containing u. This requires $O(n \cdot m)$ space where m is the maximum set size, and answers queries in $O(t)$ time where t is the output size.

To prove optimality, we establish a lower bound. Consider any data structure that correctly answers 1-lopsided queries. For any element $q \in U$, the data structure must be able to report all sets containing q.

Suppose there exist two distinct elements $q_1, q_2 \in U$ such that the data structure stores the same information for both elements. Then there must exist a collection of sets where q_1 appears in set S_i but q_2 does not (or vice versa). In this case, the data structure cannot distinguish between queries $Q_1 = \{q_1\}$ and $Q_2 = \{q_2\}$, leading to incorrect answers.

Therefore, the data structure must maintain distinct information for each element about which sets contain it. This requires $\Omega(\sum_{i=1}^{n} |S_i|)$ space in the worst case, which matches the space complexity of the trivial solution.

For query time, any correct data structure must output all t sets that intersect the query element, requiring $\Omega(t)$ time. The trivial solution achieves $O(t)$ query time by direct lookup.

Since the trivial solution achieves optimal space complexity $O(\sum_{i=1}^{n} |S_i|)$ and optimal query time $O(t)$ where t is the output size, and we have shown these bounds are tight, no improvement is possible for $k = 1$. □

L.4.4 Gemini

Proof. Let $U = \{1, \ldots, n\}$ be the universe of elements. The k-lopsided set intersection problem generally involves preprocessing a collection of sets or a single set $S \subseteq U$ to answer intersection queries with a query set Q. For the specific case of $k = 1$, the problem reduces to the fundamental Set Membership problem (or equivalently, the intersection of a stored set S with a singleton query set $Q = \{q\}$). We show that the space-time complexity cannot be improved beyond the trivial solution by a reduction from the INDEXING problem.

Consider the INDEXING problem in the one-way communication model. Alice possesses a string $x \in \{0, 1\}^n$, and Bob possesses an index $i \in \{1, \ldots, n\}$. Bob wishes to determine the value x_i. Alice sends a single message $M(x)$ to Bob, from which Bob must compute x_i.

We map the INDEXING problem to the $k = 1$ set intersection problem as follows: 1. Alice constructs a set $S \subseteq U$ such that $j \in S$ if and only if $x_j = 1$. This set S constitutes the data structure. 2. Bob's query corresponds to the singleton set $Q = \{i\}$. 3. The intersection $S \cap Q$ is non-empty if and only if $i \in S$, which is equivalent to $x_i = 1$.

It is a well-established result in communication complexity that the randomized one-way communication complexity of the INDEXING problem is $\Omega(n)$. That is, to allow Bob to correctly determine x_i with constant probability for any i, the message $M(x)$ (and thus the size of the data structure representing S) must contain $\Omega(n)$ bits of information.

The trivial solution for this problem is to store the characteristic vector of S, which requires n bits of space and allows for $O(1)$ query time. Since the information-theoretic lower bound is $\Omega(n)$ bits, any data structure solving the $k = 1$ lopsided set intersection problem must use space linear in n. Therefore, it is impossible to improve upon the space complexity of the trivial solution asymptotically. □

L.5 Problem 11.5

L.5.1 Problem Statement

Theorem L.5. *Any data structure solving the graph connectivity problem with error at most* $1/3$ *for* $m \geq n + 1$ *operations satisfies*

$$t_q \cdot \log t_u w \geq \Omega(\log n).$$

Lemma L.6.

$$\mathrm{H}(R|D) \geq 1 - \frac{7t_u w}{k}.$$

L.5.2 GPT

Proof. We are tasked with proving that any data structure solving the graph connectivity problem with error at most $\frac{1}{3}$ for $m \geq n + 1$ operations satisfies the inequality

$$t_q \cdot \log t_u w \geq \Omega(\log n).$$

Step 1: Understanding the Problem

Let the data structure be designed to solve the graph connectivity problem. This problem involves determining whether there is a path between two vertices in a graph after performing a series of operations. The problem specifies that the data structure must have an error probability of at most $\frac{1}{3}$ and supports at least $m \geq n + 1$ operations. Here, t_q and t_u represent the time complexities of query and update operations, respectively, and w is a parameter related to the weight of operations. We aim to prove that the time complexity of these operations satisfies the inequality above.

Step 2: Use of Claim 11.19

We now invoke Claim 11.19, which states that

$$H(R|D) \geq 1 - \frac{7t_u w}{k}.$$

This claim provides a lower bound on the entropy of some random variable R conditioned on a data structure D. The exact interpretation of $H(R|D)$ depends on the specific entropy model used for the data structure, but in this context, it is related to the amount of uncertainty in the data structure's operations. The parameters t_u, w, and k are related to the time complexity and error bounds of the data structure, and this bound is crucial in establishing the overall lower bound for the time complexity.

Step 3: Entropy and Lower Bounds

The claim gives us a lower bound on $H(R|D)$, which is a measure of uncertainty or entropy in the system. Intuitively, this entropy is tied to the information-theoretic lower bound on the time complexity of the data structure. In particular, the entropy bound implies that for a data structure to operate with error probability at most $\frac{1}{3}$, there must be sufficient time complexity to account for the uncertainty represented by $H(R|D)$.

Thus, we expect the time complexity of the query and update operations to scale logarithmically with the size of the graph, as entropy generally implies a lower bound on the complexity of algorithms that resolve uncertain or probabilistic information.

Step 4: Relating the Time Complexity to the Entropy Bound

Given the entropy bound, we use information-theoretic arguments to show that the product $t_q \cdot \log t_u w$ must be at least $\Omega(\log n)$. The key idea is that the entropy of a random variable (in this case, related to the graph connectivity problem) imposes a lower bound on the time complexity of solving the problem. More specifically, for a graph with n vertices and m operations, the information-theoretic lower bound on the complexity of the problem scales logarithmically with the number of vertices, which is reflected in the $\Omega(\log n)$ term.

To formalize this, observe that the data structure must process enough information to distinguish between different possible graph configurations. Since the data structure operates with a bounded error rate, the time complexity must be large enough to ensure that the probability of error remains sufficiently small, which leads to the lower bound on the time complexities of the operations.

Step 5: Conclusion

From the information-theoretic lower bound and the entropy bound given by Claim 11.19, we conclude that for the data structure to solve the graph connectivity problem with the specified error probability, it must satisfy the inequality

$$t_q \cdot \log t_u w \geq \Omega(\log n).$$

Thus, the proof is complete. □

L.5.3 Claude

Proof. We prove the theorem by establishing a communication complexity lower bound for the graph connectivity problem.

Consider a hard distribution over sequences of operations. Let k denote the number of queries in the operation sequence, and assume $k \geq \log n$ (otherwise the bound is trivial). We construct a communication game where Alice receives some information about the updates and Bob must answer queries.

Let R denote a random variable representing which query receives a particular answer (say, whether two specific vertices are connected), and let D denote the data structure state that Bob observes.

By the lemma, we have

$$\mathrm{H}(R|D) \geq 1 - \frac{7t_u w}{k}.$$

Since the data structure has error probability at most $1/3$, Bob must be able to answer queries correctly with probability at least $2/3$. By Fano's inequality, if Bob can predict R from D with error probability at most $1/3$, then

$$\mathrm{H}(R|D) \leq h(1/3) + (1/3)\log k \leq 1 + \frac{\log k}{3},$$

where $h(\cdot)$ is the binary entropy function and $h(1/3) \leq 1$.

Combining with the lemma:

$$1 - \frac{7t_u w}{k} \leq 1 + \frac{\log k}{3}.$$

This gives us

$$-\frac{7t_u w}{k} \leq \frac{\log k}{3},$$

which is always satisfied. We need a tighter analysis.

The key observation is that the information about R must flow through the queries. Each query can extract at most $t_q \log(t_u w)$ bits of information from the data structure (since the query time is t_q and each memory access reveals $\log(t_u w)$ bits about which of the possible $t_u w$ memory locations were written).

Over k queries, the total information extracted is at most $k \cdot t_q \log(t_u w)$ bits. For the data structure to succeed, we need

$$\mathrm{H}(R) - \mathrm{H}(R|D) \leq k \cdot t_q \log(t_u w).$$

Since R is chosen from k possibilities, $\mathrm{H}(R) \leq \log k$. Using the lemma with $\mathrm{H}(R|D) \geq 1 - \frac{7t_u w}{k}$:

$$\log k - \left(1 - \frac{7t_u w}{k}\right) \leq k \cdot t_q \log(t_u w).$$

For appropriately chosen $k = \Theta(\log n)$ in the hard distribution, and noting that $t_u w \leq k$ for non-trivial bounds, we have $\frac{7t_u w}{k} = O(1)$. Thus:

$$\log k - O(1) \leq k \cdot t_q \log(t_u w).$$

Since $k = \Theta(\log n)$, we have $\log k = \Theta(\log \log n)$. However, the hard distribution is constructed such that $k = \Theta(\log n)$ queries are needed, giving:

$$\Omega(\log n) \leq k \cdot t_q \log(t_u w) = \Theta(\log n) \cdot t_q \log(t_u w).$$

Dividing both sides by $\log n$:

$$\Omega(1) \leq t_q \log(t_u w).$$

More carefully, the construction ensures that each query must extract $\Omega(\log n/k) = \Omega(1)$ bits when $k = \Theta(\log n)$, yielding:

$$t_q \log(t_u w) \geq \Omega\left(\frac{\log n}{k}\right) = \Omega(1).$$

Multiplying by $k = \Theta(\log n)$ to account for the amortization over all queries:

$$t_q \log(t_u w) \geq \Omega(\log n).$$

□

L.5.4 Gemini

Proof. We establish the lower bound by considering a hard input distribution based on the information transfer tree technique. We construct a complete k-ary tree of height h, where the leaves correspond to the n vertices of the graph. The relationship between the number of vertices, the branching factor, and the height is given by $n = k^h$, which implies:

$$h = \frac{\log n}{\log k}.$$

The connectivity problem on this structure requires the data structure to maintain information about paths from the leaves to the root. Consider the decision problem at a single node in the tree, where the algorithm must identify the correct child pointer among k possibilities based on the updates. Let R be the random

variable representing the correct edge (or bit of information) at this node, and let D denote the state of the data structure after a sequence of updates.

We invoke the provided Lemma, which bounds the conditional entropy of R given the data structure D:

$$\mathrm{H}(R|D) \geq 1 - \frac{7t_u w}{k}.$$

This inequality quantifies the information gap: if the branching factor k is large relative to the update capacity $t_u w$, the data structure cannot store enough information to resolve R with certainty.

We choose the branching factor k such that the entropy remains bounded away from zero. Specifically, let $k = \lceil 14t_u w \rceil$. Substituting this choice into the Lemma yields:

$$\mathrm{H}(R|D) \geq 1 - \frac{7t_u w}{14t_u w} = 1 - \frac{1}{2} = \frac{1}{2}.$$

Given that the algorithm must solve the connectivity problem with an error probability of at most $1/3$, Fano's inequality implies that the conditional entropy of the result given the query probes must be strictly less than 1 (specifically, less than the binary entropy function $h(1/3) \approx 0.92$).

Since the data structure D (representing the information stored during updates) retains high entropy ($\geq 1/2$) regarding the decision at each level, the query algorithm cannot rely solely on the stored data to "skip" levels of the tree. To reduce the error probability to $1/3$, the query must perform probes at each level of the hierarchy to resolve the path. Therefore, the query time t_q is lower-bounded by the height of the tree:

$$t_q = \Omega(h).$$

Substituting the expression for h:

$$t_q \geq \Omega\left(\frac{\log n}{\log k}\right).$$

Substituting our choice of $k = \Theta(t_u w)$:

$$t_q \geq \Omega\left(\frac{\log n}{\log(t_u w)}\right).$$

Multiplying both sides by $\log(t_u w)$, we obtain:

$$t_q \cdot \log(t_u w) \geq \Omega(\log n).$$

□

M Problems in Chapter 12

M.1 Problem 12.1

M.1.1 Problem Statement

Theorem M.1. *Let $P = \{x : Ax \leq 0\}$ be a nonempty polytope that is contained in the halfspace $H = \{x : hx \leq c\}$.*

1. Prove that if K is the conical hull of finitely many points, and x is a point outside K, then there is a hyperplane separating x and K. Namely, x lies on one side of the hyperplane, and K on the other side (except the origin).

2. Let $A_1, \ldots, A_n$ be the rows of A. Prove that h is in the conical hull of the n points $A_1, \ldots, A_n$.

3. Conclude that Fact 12.1 is true in this case—there is $u \geq 0$ such that $uA = h$ and $ub \leq c$.

M.1.2 GPT

Proof. 1. Let K be the conical hull of finitely many points, say $K = \text{cone}\{x_1, x_2, \ldots, x_m\}$, and suppose x is a point outside K. We aim to prove that there exists a hyperplane that separates x from K, i.e., there exists a hyperplane $H = \{y : hy \leq d\}$ such that $x \notin H$ and $K \subseteq H^c$ (the complement of H), with the additional condition that the origin does not lie on the hyperplane.

By the Separating Hyperplane Theorem, since $x \notin K$, there exists a hyperplane separating x and K. Specifically, there exists a hyperplane with the equation $hx = d$ for some vector $h \in \mathbb{R}^n$ and some scalar $d \in \mathbb{R}$, such that x lies on one side of the hyperplane, and K lies on the other side. This proves the statement.

2. Let the rows of the matrix A be denoted by $A_1, A_2, \ldots, A_n$. We are tasked with proving that h lies in the conical hull of the points $A_1, A_2, \ldots, A_n$. To do so, we will show that there exists a vector $u \geq 0$ (component-wise) such that $h = \sum_{i=1}^n u_i A_i$.

Since $P = \{x : Ax \leq 0\}$ is a polytope contained in the halfspace $H = \{x : hx \leq c\}$, we know that h satisfies the inequality $hx \leq c$ for all $x \in P$. Thus, for each x satisfying $Ax \leq 0$, we must have $hx \leq c$. The hyperplane defined by h is consistent with this inequality, and since P is the polytope defined by $Ax \leq 0$, the vector h can be written as a non-negative combination of the rows $A_1, A_2, \ldots, A_n$ by the properties of linear programming and convexity. Therefore, there exists a vector $u \geq 0$ such that $h = \sum_{i=1}^n u_i A_i$.

3. From the previous result, we conclude that there exists a vector $u \geq 0$ such that $h = \sum_{i=1}^n u_i A_i$. Furthermore, since h is a hyperplane separating the polytope P from the halfspace H, it must also satisfy the condition $ub \leq c$ for some b corresponding to the defining inequality of the halfspace. This completes the proof that there exists a vector $u \geq 0$ such that $uA = h$ and $ub \leq c$.

□

M.1.3 Claude

Proof. We prove each part in turn.

Part 1: Let $K = \text{cone}(v_1, \ldots, v_m)$ be the conical hull of finitely many points $v_1, \ldots, v_m$, and suppose $x \notin K$. We need to show there exists a hyperplane separating x from K.

Consider the optimization problem: minimize $\|y - x\|^2$ subject to $y \in K$. Since K is a closed convex cone and $x \notin K$, this problem has a unique optimal solution $y^* \in K$ with $y^* \neq x$. By the optimality conditions for convex optimization, the vector $x - y^*$ is orthogonal to K at y^*. More precisely, for all $z \in K$, we have $(x - y^*)^T(z - y^*) \leq 0$.

Since K is a cone, for any $z \in K$ and $\lambda \geq 0$, we have $\lambda z \in K$. Taking $z \in K$ and letting $\lambda \to \infty$ in the inequality $(x - y^*)^T(\lambda z - y^*) \leq 0$, we obtain $(x - y^*)^T z \leq 0$ for all $z \in K$.

Also, $(x - y^*)^T y^* \leq 0$. Since $y^* \in K$ and $0 \in K$, and $(x - y^*)^T \cdot 0 = 0$, we have $(x - y^*)^T y^* \leq 0$.

Now, $(x - y^*)^T x = (x - y^*)^T(x - y^*) + (x - y^*)^T y^* = \|x - y^*\|^2 + (x - y^*)^T y^* > 0$ since $x \neq y^*$ and $(x - y^*)^T y^* \leq 0$.

Therefore, the hyperplane $\{w : (x - y^*)^T w = 0\}$ separates x from K, with $(x - y^*)^T x > 0$ and $(x - y^*)^T z \leq 0$ for all $z \in K$.

Part 2: We prove that h is in the conical hull of $A_1, \ldots, A_n$, where A_i are the rows of A.

Suppose, for contradiction, that $h \notin \text{cone}(A_1, \ldots, A_n)$. By Part 1, there exists a hyperplane separating h from $\text{cone}(A_1, \ldots, A_n)$. That is, there exists a vector x^* such that $h^T x^* > 0$ and $A_i^T x^* \leq 0$ for all $i = 1, \ldots, n$.

The condition $A_i^T x^* \leq 0$ for all i means $(Ax^*)_i \leq 0$ for all i, which implies $Ax^* \leq 0$. Therefore, $x^* \in P$.

Since $x^* \in P$ and $P \subseteq H$, we have $x^* \in H$, which means $h^T x^* \leq c$.

However, we also have $h^T x^* > 0$ from the separation property. Since P is nonempty, there exists $x_0 \in P$ with $Ax_0 \leq 0$. For any $\lambda > 0$, we have $A(\lambda x^*) = \lambda A x^* \leq 0$, so $\lambda x^* \in P \subseteq H$, giving $h^T(\lambda x^*) \leq c$, or $\lambda h^T x^* \leq c$.

Since $h^T x^* > 0$, taking $\lambda \to \infty$ yields a contradiction. Therefore, $h \in \text{cone}(A_1, \ldots, A_n)$.

Part 3: Since $h \in \text{cone}(A_1, \ldots, A_n)$, there exist $u_1, \ldots, u_n \geq 0$ such that $h = \sum_{i=1}^n u_i A_i = u^T A$, where $u = (u_1, \ldots, u_n)^T \geq 0$. This gives $uA = h$.

To show $ub \leq c$, note that for any $x \in P$, we have $Ax \leq 0$, which implies $u^T Ax \leq 0$ since $u \geq 0$. Thus $h^T x \leq 0$ for all $x \in P$. However, we need to verify the condition $ub \leq c$ where b should be interpreted appropriately. Given that $P \subseteq H$ and the constraint structure, for all $x \in P$, we have $hx \leq c$. Since P is nonempty and contains points arbitrarily far in directions satisfying $Ax \leq 0$, and $h = uA$, the condition $ub \leq c$ follows from the containment $P \subseteq H$ when properly interpreted in the context of Fact 12.1. □

M.1.4 Gemini

Proof. We address the three parts of the theorem sequentially.

First, let K be the conical hull of finitely many points. K is a closed convex cone. Let x be a point such that $x \notin K$. Since K is a non-empty closed convex set, there exists a unique projection $y \in K$ minimizing the Euclidean distance $\|x - y\|$. Let $w = x - y$. Since $x \notin K$, $w \neq 0$. By the variational characterization of the projection, for all $z \in K$, we have $(z - y) \cdot (x - y) \leq 0$, which can be written as $(z - y) \cdot w \leq 0$, or $z \cdot w \leq y \cdot w$.

Since K is a cone, for any $z \in K$, $2z \in K$ and $0 \in K$. Substituting $z = 2y$ into the inequality yields $2y \cdot w \leq y \cdot w$, implying $y \cdot w \leq 0$. Substituting $z = 0$ yields $-y \cdot w \leq 0$, implying $y \cdot w \geq 0$. Therefore, $y \cdot w = 0$. The inequality simplifies to $z \cdot w \leq 0$ for all $z \in K$. Furthermore, considering the point x, we have $x \cdot w = (y + w) \cdot w = y \cdot w + \|w\|^2 = \|w\|^2 > 0$. Thus, the hyperplane $H_w = \{v : v \cdot w = 0\}$ separates x and K, with K contained in the halfspace $\{v : v \cdot w \leq 0\}$ and x contained in the open halfspace $\{v : v \cdot w > 0\}$. The origin satisfies $0 \cdot w = 0$, lying on the hyperplane.

Second, let $A_1, \ldots, A_n$ be the rows of the matrix A. Let $C = \text{cone}(A_1, \ldots, A_n)$ be the conical hull of these rows. We are given the polytope $P = \{x : Ax \leq 0\}$ and the halfspace $H = \{x : hx \leq c\}$ such that $P \subseteq H$. Since P is defined by a homogeneous system of inequalities, P is a cone containing the origin. The condition $P \subseteq H$ implies that for all $x \in P$, $hx \leq c$. Since P is a cone, if there existed any $x \in P$ such that $hx > 0$, then for sufficiently large $\lambda > 0$, the point $\lambda x \in P$ would satisfy $h(\lambda x) > c$, contradicting the containment. Thus, we must have $hx \leq 0$ for all $x \in P$.

We proceed by contradiction to prove $h \in C$. Suppose $h \notin C$. By the result in part 1, there exists a separating vector y such that $v \cdot y \leq 0$ for all $v \in C$ and $h \cdot y > 0$. Since each row A_i is in C, we have $A_i \cdot y \leq 0$ for all $i = 1, \ldots, n$. In matrix notation, this implies $Ay \leq 0$. By the definition of P, this means $y \in P$. However, we established that for all $x \in P$, $hx \leq 0$. The separation condition $h \cdot y > 0$ (equivalently $hy > 0$) contradicts this necessary condition. Therefore, the assumption $h \notin C$ is false, and h must be in the conical hull of the rows of A.

Third, since h is in the conical hull of the rows $A_1, \ldots, A_n$, by definition there exist non-negative coefficients $u_1, \ldots, u_n \geq 0$ such that $h = \sum_{i=1}^{n} u_i A_i$. In matrix notation, let $u = (u_1, \ldots, u_n)$, then $u \geq 0$ and $uA = h$.

To conclude Fact 12.1, we must show $ub \leq c$. In the statement of the theorem, P is defined as $\{x : Ax \leq 0\}$, which corresponds to the standard form $Ax \leq b$ with $b = 0$. Consequently, $ub = u \cdot 0 = 0$. Since P is nonempty (it contains the origin) and $P \subseteq H$, the origin must satisfy the inequality defining H. Thus, $h \cdot 0 \leq c$, which implies $0 \leq c$. Therefore, we have $ub = 0 \leq c$. We have constructed $u \geq 0$ such that $uA = h$ and $ub \leq c$. □

M.2 Problem 12.2

M.2.1 Problem Statement

Lemma M.2. *Farkas's Lemma states for every convex set $S \subseteq \mathbb{R}^d$, and every point $x \notin S$, there is a halfspace containing S but not containing x. Use the lemma to prove Fact 12.1. Hint: Consider the $(n + 1) \times (d + 1)$ matrix A' obtained by adding b as a column of A and adding one more row: $(0, \ldots, 0, -1)$. Apply Farkas's Lemma to the conical hull of the rows of A' and a point that depends on the given halfspace.*

M.2.2 GPT

Proof. We are tasked with proving Fact 12.1 using Farkas's Lemma. Let $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$ be the matrix and vector associated with the given linear system. The conical hull of the rows of A is denoted by

cone(A), which is the set of all non-negative linear combinations of the rows of A. We aim to show that the halfspace defined by the inequality $Ax \leq b$ contains a certain set while not containing a specific point, using Farkas's Lemma.

Let us construct the $(n+1) \times (d+1)$ matrix A' by appending b as an additional column to A and adding one more row $(0, \ldots, 0, -1)$ as follows:

$$A' = \begin{pmatrix} A & b \\ 0 & -1 \end{pmatrix}.$$

Next, we apply Farkas's Lemma to the conical hull of the rows of A'. Let $y = (y_1, \ldots, y_n, y_{n+1})^T$ be a vector such that $y \in \mathbb{R}^{n+1}$ and satisfies the condition $y^T A' = 0$ with $y \geq 0$. The condition $y^T A' = 0$ expands to:

$$y_1 \cdot A_1 + \cdots + y_n \cdot A_n + y_{n+1} \cdot (0, \ldots, 0, -1) = 0,$$

which simplifies to:

$$y_1 A_1 + \cdots + y_n A_n - y_{n+1} = 0.$$

Since y_{n+1} is non-negative, we have $y_{n+1} = 0$ and thus the equation becomes:

$$y_1 A_1 + \cdots + y_n A_n = 0.$$

Thus, the vector y lies in the null space of A. We now consider the second part of the lemma: if there exists a point x such that $Ax \leq b$, then there must be a corresponding dual solution that satisfies the above equation. Therefore, by Farkas's Lemma, we can conclude that the halfspace defined by $Ax \leq b$ contains the conical hull of the rows of A' while not containing the point x, as desired.

Thus, we have shown that the halfspace $Ax \leq b$ contains the conical hull of the rows of A but does not contain the point x, completing the proof. □

M.2.3 Claude

Proof. We prove Fact 12.1 using Farkas's Lemma. Recall that Fact 12.1 states: For a system $Ax \leq b$ where $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$, exactly one of the following holds:

(i) There exists $x \in \mathbb{R}^d$ such that $Ax \leq b$.

(ii) There exists $y \in \mathbb{R}^n$ with $y \geq 0$, $y^T A = 0$, and $y^T b < 0$.

Following the hint, we construct the $(n+1) \times (d+1)$ matrix A' by augmenting A with column b and adding row $(0, \ldots, 0, -1)$:

$$A' = \begin{pmatrix} A & b & & \\ 0 & \cdots & 0 & -1 \end{pmatrix}$$

Let S denote the conical hull of the rows of A', that is,

$$S = \left\{ \sum_{i=1}^{n+1} \lambda_i r_i : \lambda_i \geq 0 \right\}$$

where $r_1, \ldots, r_n$ are the first n rows of A' (corresponding to (A_i, b_i) for each row i of the original system), and $r_{n+1} = (0, \ldots, 0, -1)$.

Consider the point $p = (0, \ldots, 0, 1) \in \mathbb{R}^{d+1}$.

We now analyze two cases:

Case 1: Suppose $p \in S$. Then there exist $\lambda_1, \ldots, \lambda_{n+1} \geq 0$ such that

$$\sum_{i=1}^{n} \lambda_i (A_i, b_i) + \lambda_{n+1}(0, \ldots, 0, -1) = (0, \ldots, 0, 1)$$

Examining the first d coordinates: $\sum_{i=1}^{n} \lambda_i A_i = 0$.

Examining the last coordinate: $\sum_{i=1}^{n} \lambda_i b_i - \lambda_{n+1} = 1$.

Since $\lambda_{n+1} \geq 0$, we have $\sum_{i=1}^{n} \lambda_i b_i \geq 1 > 0$.

Let $y = (\lambda_1, \ldots, \lambda_n)^T$. Then $y \geq 0$, $y^T A = 0$, and $y^T b > 0$. This means alternative (ii) fails (since we need $y^T b < 0$), so alternative (i) must hold.

Case 2: Suppose $p \notin S$. By Farkas's Lemma, there exists a halfspace containing S but not p. That is, there exists $c \in \mathbb{R}^{d+1}$ such that $\langle c, s \rangle \leq 0$ for all $s \in S$ and $\langle c, p \rangle > 0$.

Write $c = (c_1, \ldots, c_d, c_{d+1})$. Since S is a cone and contains the origin, we have $\langle c, r_i \rangle \leq 0$ for all rows r_i of A'.

For $i = 1, \ldots, n$: $\langle c, (A_i, b_i) \rangle = \langle (c_1, \ldots, c_d), A_i \rangle + c_{d+1} b_i \leq 0$.

For $i = n+1$: $\langle c, (0, \ldots, 0, -1) \rangle = -c_{d+1} \leq 0$, so $c_{d+1} \geq 0$.

Also, $\langle c, p \rangle = c_{d+1} > 0$, so $c_{d+1} > 0$.

Let $x = -\frac{1}{c_{d+1}}(c_1, \ldots, c_d)$. Then for each $i = 1, \ldots, n$:

$$\langle A_i, x \rangle = -\frac{1}{c_{d+1}} \langle A_i, (c_1, \ldots, c_d) \rangle \leq \frac{1}{c_{d+1}}(-c_{d+1} b_i) = -b_i$$

Thus $A_i x \leq b_i$ for all i, which means $Ax \leq b$. Therefore alternative (i) holds.

Since exactly one of the two cases occurs, exactly one of alternatives (i) and (ii) holds. □

M.2.4 Gemini

Proof. We prove Fact 12.1, which is the algebraic form of Farkas's Lemma (Theorem of the Alternative): The system of linear inequalities $Ax \leq b$ has a solution $x \in \mathbb{R}^d$ if and only if there is no vector $y \in \mathbb{R}^n$ with $y \geq 0$ such that $y^T A = 0$ and $y^T b < 0$.

To prove this, we first establish the condition for the system $Ax \leq -b$. Let A' be the $(n+1) \times (d+1)$ matrix constructed as described in the hint: the first n rows are formed by appending b_i to the row a_i, and the $(n+1)$-th row is $(0, \ldots, 0, -1)$. Let $r_1, \ldots, r_{n+1}$ denote the rows of A':

$$r_i = (a_i, b_i) \in \mathbb{R}^{d+1} \quad \text{for } i = 1, \ldots, n, \quad \text{and} \quad r_{n+1} = (\mathbf{0}, -1).$$

Let C be the conical hull of these rows:

$$C = \text{cone}(r_1, \ldots, r_{n+1}) = \left\{ \sum_{i=1}^{n+1} \lambda_i r_i \;\middle|\; \lambda_i \geq 0 \right\}.$$

C is a convex set. Consider the point $p = (\mathbf{0}, 1) \in \mathbb{R}^{d+1}$. By the geometric Farkas's Lemma provided, either $p \in C$ or there exists a halfspace containing C but not p.

Case 1: $p \in C$**.** There exist coefficients $\lambda_1, \ldots, \lambda_{n+1} \geq 0$ such that $p = \sum_{i=1}^{n+1} \lambda_i r_i$. Writing this component-wise:

$$\mathbf{0} = \sum_{i=1}^{n} \lambda_i a_i \quad \text{and} \quad 1 = \sum_{i=1}^{n} \lambda_i b_i - \lambda_{n+1}.$$

Let $y = (\lambda_1, \ldots, \lambda_n)^T$. Then $y \geq 0$ and $y^T A = 0$. From the second equation, $y^T b = 1 + \lambda_{n+1} \geq 1 > 0$. If there existed a solution x to $Ax \leq -b$, then $y^T A x = 0$. However, $Ax \leq -b$ and $y \geq 0$ implies $y^T A x \leq y^T(-b) = -y^T b$. This leads to $0 \leq -y^T b$, or $y^T b \leq 0$, contradicting $y^T b > 0$. Thus, if $p \in C$, the system $Ax \leq -b$ is infeasible.

Case 2: $p \notin C$**.** By the Lemma, there exists a halfspace containing C but not p. Since C is a cone containing the origin, the separating hyperplane must pass through the origin. Thus, there exists a vector $h = (w, \mu) \in \mathbb{R}^d \times \mathbb{R}$ such that:

$$h^T z \leq 0 \quad \forall z \in C \quad \text{and} \quad h^T p > 0.$$

The condition $h^T p > 0$ implies $(w, \mu) \cdot (\mathbf{0}, 1) = \mu > 0$. We can scale h such that $\mu = 1$. The condition $h^T r_i \leq 0$ for all i implies: 1. For $i = n+1$: $(w, 1) \cdot (\mathbf{0}, -1) = -1 \leq 0$ (always true). 2. For $i = 1, \ldots, n$: $(w, 1) \cdot (a_i, b_i) = a_i w + b_i \leq 0 \implies a_i w \leq -b_i$. Thus, w is a solution to the system $Ax \leq -b$.

We have shown that $Ax \leq -b$ is feasible if and only if there does not exist $y \geq 0$ such that $y^T A = 0$ and $y^T b > 0$. To prove Fact 12.1 for the system $Ax \leq b$, we replace b with $-b$ in the derived statement. The system $Ax \leq b$ is feasible if and only if there does not exist $y \geq 0$ such that $y^T A = 0$ and $y^T(-b) > 0$, which is equivalent to $y^T b < 0$. □

M.3 Problem 12.3

M.3.1 Problem Statement

Theorem M.3. *The nonnegative rank of the slack matrix of a regular 2^k-sided polygon in the plane is at most $O(k)$.*

M.3.2 GPT

Proof. Let P be the set of vertices of a regular 2^k-sided polygon inscribed in the unit circle in the plane. The slack matrix of P is defined as the matrix whose rows correspond to the inequalities defining the convex hull of P, and whose columns correspond to the coordinates of the points of P.

We aim to prove that the nonnegative rank of the slack matrix of P is at most $O(k)$.

Consider the matrix S of size $2^k \times 2^k$, where each entry of S corresponds to the slack of a linear inequality satisfied by the points of the polygon. The rank of S is the dimension of the column space, which can be understood in terms of the linear dependence of the constraints. Since the number of vertices of a regular 2^k-sided polygon is 2^k, the number of linear constraints on these vertices is bounded above by a constant multiple of k due to the symmetry of the regular polygon.

The crucial observation here is that the slack matrix for a regular 2^k-sided polygon is highly structured. Specifically, each vertex of the polygon is related to others through rotational symmetries, meaning that the rows of the slack matrix corresponding to these vertices are not independent. Instead, they exhibit a pattern of dependence which leads to a rank that grows only linearly with respect to k.

To formalize this, we examine the rank of the matrix formed by the inequalities corresponding to the geometric properties of a regular 2^k-sided polygon. The essential idea is that the number of independent constraints needed to describe the geometry of the polygon is at most proportional to the number of distinct angles between adjacent vertices, which is related to k. Since the number of these constraints is $O(k)$, the rank of the slack matrix is also $O(k)$.

Thus, the nonnegative rank of the slack matrix, which is bounded by the rank of the matrix, must also be at most $O(k)$.

Therefore, the nonnegative rank of the slack matrix of a regular 2^k-sided polygon is at most $O(k)$, as required. □

M.3.3 Claude

Proof. Let P be a regular 2^k-sided polygon in the plane. We denote the vertices of P as $v_0, v_1, \ldots, v_{2^k-1}$ arranged in cyclic order, and the edges (facets) as $f_0, f_1, \ldots, f_{2^k-1}$ where edge f_i connects vertices v_i and $v_{i+1 \pmod{2^k}}$.

The slack matrix S of P is a $2^k \times 2^k$ matrix where entry S_{ij} represents the distance from vertex v_i to the line containing edge f_j. For a regular polygon, this distance is nonnegative and equals zero if and only if vertex v_i lies on edge f_j.

Without loss of generality, place the regular 2^k-sided polygon centered at the origin with vertices at positions $v_i = (\cos(\theta_i), \sin(\theta_i))$ where $\theta_i = \frac{2\pi i}{2^k}$ for $i = 0, 1, \ldots, 2^k - 1$.

The edge f_j connecting v_j and v_{j+1} can be described by a linear equation. The signed distance from vertex v_i to the line containing edge f_j is given by a linear function of the coordinates of v_i.

For a regular 2^k-sided polygon, the slack matrix exhibits strong symmetry properties. Specifically, S is a circulant-like matrix where S_{ij} depends only on $i - j \pmod{2^k}$.

We exploit the hierarchical structure of the regular 2^k-sided polygon. The key observation is that we can decompose the polygon recursively using the binary representation of k. At each level of the hierarchy, we can express the slack values using a bounded number of basis functions.

Define $r = \lceil 3k + 2 \rceil$. We construct nonnegative matrices $A \in \mathbb{R}_+^{2^k \times r}$ and $B \in \mathbb{R}_+^{r \times 2^k}$ such that $S = AB$.

The construction proceeds as follows. For each level $\ell = 0, 1, \ldots, k-1$ in the binary hierarchy, we introduce $O(1)$ basis vectors that capture the geometric structure at that scale. Specifically, we use:

(1) Basis vectors corresponding to the angular position of vertices, which can be represented using trigonometric functions $\cos(2^\ell \theta_i)$ and $\sin(2^\ell \theta_i)$ for $\ell = 0, 1, \ldots, k-1$.

(2) Basis vectors that encode the distance relationships between vertices and edges at different scales of the hierarchy.

(3) Constant basis vectors to handle the offset terms in the distance calculations.

For each level ℓ, we require at most 3 basis vectors, giving a total of at most $3k + 2$ basis vectors across all levels.

The matrix A has rows indexed by vertices, where the (i, ℓ)-th block of row i contains the evaluations of the basis functions at vertex v_i for level ℓ.

The matrix B has columns indexed by edges, where the (ℓ, j)-th block of column j contains the coefficients that combine the basis functions to produce the slack values for edge f_j.

By the symmetry and regularity of the polygon, all entries of A and B can be chosen to be nonnegative. The factorization $S = AB$ follows from the linearity of the distance function and the completeness of the hierarchical basis.

Therefore, the nonnegative rank of S satisfies $\text{rank}_+(S) \leq r = O(k)$. □

M.3.4 Gemini

Theorem M.4. *The nonnegative rank of the slack matrix of a regular 2^k-sided polygon in the plane is at most $O(k)$.*

Proof. Let $n = 2^k$ and let P_n denote a regular polygon with n sides in $\mathbb{R}^2$. Let S be the slack matrix of P_n. By Yannakakis' Theorem, the nonnegative rank of S, denoted $\text{rank}_+(S)$, is equal to the extension complexity of P_n, denoted $xc(P_n)$. The extension complexity $xc(P_n)$ is the minimum number of facets in a polytope Q of higher dimension such that P_n is a linear projection of Q. We proceed to construct an extended formulation for P_n of size $O(k)$.

We construct the regular 2^k-gon recursively. Let R_θ denote the rotation matrix in $\mathbb{R}^2$ by angle θ, and let $J = R_{\pi/2} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$.

Consider the sequence of polytopes defined as follows. Let Q_2 be the square (regular 4-gon) defined by:

$$Q_2 = \{x \in \mathbb{R}^2 : \|x\|_\infty \leq 1\} = \{x \in \mathbb{R}^2 : -1 \leq x_1 \leq 1, -1 \leq x_2 \leq 1\}.$$

For $m \geq 2$, suppose we have an extended formulation for the regular 2^m-gon Q_m. We construct Q_{m+1} (a regular 2^{m+1}-gon) using the following geometric property: A regular 2^{m+1}-gon can be realized as the projection of a set defined by the regular 2^m-gon and auxiliary constraints.

Let $\alpha_m = \frac{\pi}{2^{m+1}}$. We define the relation between a variable $x^{(m)} \in \mathbb{R}^2$ (representing a point in Q_m) and $x^{(m+1)} \in \mathbb{R}^2$ (representing a point in Q_{m+1}) by the equation:

$$x^{(m)} = \cos(\alpha_m) x^{(m+1)} + u_m \sin(\alpha_m) J x^{(m+1)},$$

where u_m is a scalar variable satisfying $-1 \leq u_m \leq 1$.

Ben-Tal and Nemirovski (2001) established that if $x^{(m)}$ ranges over a regular 2^m-gon Q_m, then the set of points $x^{(m+1)}$ satisfying the above relation for some $u_m \in [-1, 1]$ forms a regular 2^{m+1}-gon Q_{m+1} (up to scaling). Specifically, the inclusion $x^{(m)} \in Q_m$ is equivalent to requiring $x^{(m+1)}$ to satisfy the facet inequalities of Q_{m+1}.

We define the extended formulation for the regular 2^k-gon P_n (where $n = 2^k$) by chaining these relations. Let $x^{(2)}, x^{(3)}, \ldots, x^{(k)}$ be variables in $\mathbb{R}^2$, and $u_2, \ldots, u_{k-1}$ be scalar variables. The constraints are:

$$x^{(2)} \in Q_2 \quad \text{(i.e., } \|x^{(2)}\|_\infty \leq 1\text{)},$$
$$x^{(j)} = \cos\left(\frac{\pi}{2^{j+1}}\right) x^{(j+1)} + u_j \sin\left(\frac{\pi}{2^{j+1}}\right) J x^{(j+1)}, \quad \text{for } j = 2, \ldots, k-1,$$

$$-1 \le u_j \le 1, \quad \text{for } j = 2, \dots, k-1.$$

The polytope Q in the space of variables $(x^{(2)}, \dots, x^{(k)}, u_2, \dots, u_{k-1})$ is defined by these constraints. The projection of Q onto the $x^{(k)}$ variables is the regular 2^k-gon P_n (up to a scaling factor which does not affect the combinatorial structure or the slack matrix rank).

We count the size of this formulation:

- The base constraint $x^{(2)} \in Q_2$ involves 4 linear inequalities.
- For each $j \in \{2, \dots, k-1\}$, the relation is a linear equality system (which can be substituted or represented by 2 equalities) and the bounds on u_j add 2 inequalities.
- The number of auxiliary variables u_j is $k-2$.
- The number of vector variables is $k-1$.

The total number of variables and constraints is linear in k. Specifically, the number of facets in the lifted polytope Q is $4 + 2(k-2) = 2k$.

Since there exists an extended formulation of size $O(k)$, the extension complexity of the regular 2^k-gon is $O(k)$. Consequently, the nonnegative rank of its slack matrix is at most $O(k)$. □

M.4 Problem 12.4

M.4.1 Problem Statement

Theorem M.5. *The extension complexity of any regular polygon in the plane with n facets is $O(\log n)$.*

M.4.2 GPT

Proof. Let $P_n \subset \mathbb{R}^2$ be a regular n-gon. Since extension complexity is invariant under affine isomorphisms, we may assume P_n is centered at the origin and is the convex hull of the points $v_k = (\cos(2\pi k/n), \sin(2\pi k/n))$ for $k = 0, 1, \dots, n-1$. Equivalently, P_n can be described as the intersection of n halfspaces whose outward unit normals are $u_k = (\cos(2\pi k/n), \sin(2\pi k/n))$ and whose supporting lines are tangent to the unit circle at angle $2\pi k/n$, namely

$$P_n = \{x \in \mathbb{R}^2 : \langle u_k, x\rangle \le 1 \text{ for all } k = 0, 1, \dots, n-1\}.$$

Let $S \in \mathbb{R}_+^{n\times n}$ denote the slack matrix of this facet-vertex description, i.e.,

$$S_{k\ell} = 1 - \langle u_k, v_\ell\rangle \qquad (k, \ell \in \{0, 1, \dots, n-1\}).$$

By the definition of u_k and v_ℓ we have $\langle u_k, v_\ell\rangle = \cos(2\pi(\ell - k)/n)$, hence

$$S_{k\ell} = 1 - \cos\Big(\frac{2\pi(\ell-k)}{n}\Big) = 2\sin^2\Big(\frac{\pi(\ell-k)}{n}\Big).$$

It is a standard fact that the extension complexity $\mathrm{xc}(P_n)$ equals the nonnegative rank of S, i.e. $\mathrm{xc}(P_n) = \mathrm{rank}_+(S)$.

Fix $m = \lceil \log_2 n \rceil$ so that $n \le 2^m$. Define angles $\theta_\ell = 2\pi\ell/n$ and $\phi_k = 2\pi k/n$. Then $S_{k\ell} = 1 - \cos(\theta_\ell - \phi_k)$. Consider the identity valid for all real t:

$$1 - \cos t = 2\sin^2\Big(\frac{t}{2}\Big).$$

Moreover, for any real a, b we have the product-to-sum formula

$$\sin a \, \sin b = \frac{1}{2}\big(\cos(a-b) - \cos(a+b)\big).$$

Set $t = \theta_\ell - \phi_k$ and note that

$$\sin^2\Big(\frac{t}{2}\Big) = \sin\Big(\frac{\theta_\ell - \phi_k}{2}\Big)\sin\Big(\frac{\theta_\ell - \phi_k}{2}\Big).$$

Introduce the following $2m$ functions of an angle α:

$$p_j(\alpha) = \cos(2^{j-1}\alpha), \qquad q_j(\alpha) = \sin(2^{j-1}\alpha) \qquad (j = 1, \dots, m).$$

We claim that for any angles α, β one can express $1-\cos(\alpha-\beta)$ as a sum of at most $2m$ nonnegative rank-one terms in the variables $(p_j(\alpha), q_j(\alpha))$ and $(p_j(\beta), q_j(\beta))$ after a suitable lifting, yielding $\mathrm{rank}_+(S) = O(m)$.

To formalize this, define for each $j = 1, \dots, m$ the nonnegative quantities

$$A_j(\gamma) = 1 - \cos(2^{j-1}\gamma) = 2\sin^2\Big(\frac{2^{j-1}\gamma}{2}\Big) = 2\sin^2(2^{j-2}\gamma) \geq 0,$$

with the understanding that for $j = 1$ we have $A_1(\gamma) = 1 - \cos\gamma = 2\sin^2(\gamma/2) \geq 0$. Observe the doubling identity

$$A_{j+1}(\gamma) = 1 - \cos(2^j\gamma) = 1 - (2\cos^2(2^{j-1}\gamma) - 1) = 2(1 - \cos^2(2^{j-1}\gamma)) = 2\sin^2(2^{j-1}\gamma).$$

Hence $A_{j+1}(\gamma) = 2(1 - \cos(2^{j-1}\gamma))(1 + \cos(2^{j-1}\gamma))$ because $\sin^2 t = (1 - \cos 2t)/2$ and also $1 - \cos^2 t = (1 - \cos t)(1 + \cos t)$. More precisely,

$$A_{j+1}(\gamma) = 2(1 - \cos(2^{j-1}\gamma))(1 + \cos(2^{j-1}\gamma)) = 2A_j(\gamma)\big(2 - A_j(\gamma)\big),$$

since $1 + \cos(2^{j-1}\gamma) = 2 - (1 - \cos(2^{j-1}\gamma)) = 2 - A_j(\gamma)$. Therefore the sequence $(A_j(\gamma))_{j\geq 1}$ satisfies the recursion

$$A_{j+1}(\gamma) = 2A_j(\gamma)\big(2 - A_j(\gamma)\big) \qquad (j \geq 1),$$

with $A_1(\gamma) = 1 - \cos\gamma$.

Now fix k, ℓ and write $\gamma = \theta_\ell - \phi_k \in [0, 2\pi)$. Since $n \leq 2^m$, the angle γ is an integer multiple of $2\pi/n$ and thus can be written as $\gamma = 2\pi r/n$ for some integer r with $0 \leq r \leq n - 1 < 2^m$. Let $r = \sum_{j=0}^{m-1} \varepsilon_{j+1} 2^j$ be the binary expansion of r with bits $\varepsilon_{j+1} \in \{0, 1\}$. Then

$$\gamma = \frac{2\pi}{n} \sum_{j=0}^{m-1} \varepsilon_{j+1} 2^j.$$

Define partial sums $\gamma_t = \frac{2\pi}{n} \sum_{j=0}^{t-1} \varepsilon_{j+1} 2^j$ for $t = 0, 1, \dots, m$, so that $\gamma_0 = 0$ and $\gamma_m = \gamma$. Consider the telescoping decomposition

$$A_1(\gamma) = \sum_{t=0}^{m-1} \big(A_1(\gamma_{t+1}) - A_1(\gamma_t)\big).$$

We will show that each increment $A_1(\gamma_{t+1}) - A_1(\gamma_t)$ admits a factorization into a sum of a constant number of nonnegative rank-one terms depending only on the angle γ_t and on the bit ε_{t+1}, and that γ_t itself can be propagated through the recursion using the variables p_j, q_j at level $j = t + 1$. This yields a nonnegative factorization of S with $O(m)$ factors.

For $t \in \{0, \dots, m-1\}$, either $\varepsilon_{t+1} = 0$ and then $\gamma_{t+1} = \gamma_t$, so the increment is 0, or $\varepsilon_{t+1} = 1$ and then $\gamma_{t+1} = \gamma_t + 2\pi 2^t/n$. In the latter case,

$$A_1(\gamma_{t+1}) - A_1(\gamma_t) = \big(1 - \cos(\gamma_t + \delta_t)\big) - \big(1 - \cos\gamma_t\big) = \cos\gamma_t - \cos(\gamma_t + \delta_t),$$

where $\delta_t = 2\pi 2^t/n$. Using $\cos(\gamma_t + \delta_t) = \cos\gamma_t \cos\delta_t - \sin\gamma_t \sin\delta_t$, we obtain

$$\cos\gamma_t - \cos(\gamma_t + \delta_t) = \cos\gamma_t(1 - \cos\delta_t) + \sin\gamma_t \sin\delta_t.$$

Since $0 \leq \delta_t \leq 2\pi$ and $n \geq 3$, we have $1 - \cos\delta_t \geq 0$ but $\sin\delta_t$ may have either sign. To obtain a nonnegative factorization, we split $\sin\delta_t$ into positive and negative parts. Let $\sin\delta_t = \sin_t^+ - \sin_t^-$ with $\sin_t^\pm \geq 0$ and $\sin_t^+ \sin_t^- = 0$. Then

$$\cos\gamma_t - \cos(\gamma_t + \delta_t) = \cos\gamma_t(1 - \cos\delta_t) + \sin\gamma_t \sin_t^+ - \sin\gamma_t \sin_t^- .$$

Next, decompose $\sin\gamma_t$ into its positive and negative parts: $\sin\gamma_t = \sin^+(\gamma_t) - \sin^-(\gamma_t)$ with $\sin^\pm(\gamma_t) \geq 0$ and $\sin^+(\gamma_t)\sin^-(\gamma_t) = 0$. Similarly write $\cos\gamma_t = \cos^+(\gamma_t) - \cos^-(\gamma_t)$ with $\cos^\pm(\gamma_t) \geq 0$ and $\cos^+(\gamma_t)\cos^-(\gamma_t) = 0$. Then the expression becomes a sum of at most four terms, each a product of a nonnegative function of γ_t with a nonnegative constant depending only on t:

$$\cos\gamma_t(1-\cos\delta_t) = \cos^+(\gamma_t)(1-\cos\delta_t) - \cos^-(\gamma_t)(1-\cos\delta_t),$$

$$\sin\gamma_t \sin_t^+ = \sin^+(\gamma_t)\sin_t^+ - \sin^-(\gamma_t)\sin_t^+,$$

$$-\sin\gamma_t \sin_t^- = -\sin^+(\gamma_t)\sin_t^- + \sin^-(\gamma_t)\sin_t^- .$$

Collecting, we can write

$$A_1(\gamma_{t+1}) - A_1(\gamma_t) = \sum_{s=1}^{C} f_{t,s}(\gamma_t)\, c_{t,s},$$

for some absolute constant $C \leq 8$, where each $f_{t,s}(\gamma_t) \geq 0$ is one of $\sin^\pm(\gamma_t)$ or $\cos^\pm(\gamma_t)$, and each $c_{t,s} \geq 0$ is one of $(1-\cos\delta_t)$, $\sin_t^+$, $\sin_t^-$. If $\varepsilon_{t+1} = 0$, we take all $c_{t,s} = 0$.

It remains to express each $f_{t,s}(\gamma_t)$ as a nonnegative affine function of a fixed set of lifted variables that depend linearly on $(\cos\gamma_t, \sin\gamma_t)$, and to propagate $(\cos\gamma_t, \sin\gamma_t)$ from the original data $(\cos\theta_\ell, \sin\theta_\ell)$ and $(\cos\phi_k, \sin\phi_k)$ using $O(m)$ additional variables and linear constraints.

Consider lifted variables $(a_t, b_t) \in \mathbb{R}^2$ intended to represent $(\cos\gamma_t, \sin\gamma_t)$. Initialize with $\gamma_0 = 0$, so set $a_0 = 1$ and $b_0 = 0$. For each t, define the update from (a_t, b_t) to (a_{t+1}, b_{t+1}) depending on the bit ε_{t+1} by the rotation formulas:

$$a_{t+1} = a_t\cos(\varepsilon_{t+1}\delta_t) - b_t\sin(\varepsilon_{t+1}\delta_t), \qquad b_{t+1} = a_t\sin(\varepsilon_{t+1}\delta_t) + b_t\cos(\varepsilon_{t+1}\delta_t).$$

Since $\varepsilon_{t+1} \in \{0,1\}$ and $\cos(0) = 1, \sin(0) = 0$, this update is affine-linear in (a_t, b_t) once we introduce variables encoding the choice of ε_{t+1}. We implement the choice by introducing nonnegative selector variables $\lambda_{t,0}, \lambda_{t,1} \geq 0$ with $\lambda_{t,0} + \lambda_{t,1} = 1$, and enforce

$$a_{t+1} = \lambda_{t,0}a_t + \lambda_{t,1}(a_t\cos\delta_t - b_t\sin\delta_t), \qquad b_{t+1} = \lambda_{t,0}b_t + \lambda_{t,1}(a_t\sin\delta_t + b_t\cos\delta_t).$$

This is linear in the extended variables $(a_t, b_t, a_{t+1}, b_{t+1}, \lambda_{t,0}, \lambda_{t,1})$ because $\cos\delta_t$ and $\sin\delta_t$ are fixed constants. Moreover, for each pair (k,ℓ) corresponding to a slack matrix entry, we can choose $\lambda_{t,1} = \varepsilon_{t+1}$ and $\lambda_{t,0} = 1 - \varepsilon_{t+1}$, which are feasible because $\varepsilon_{t+1} \in \{0,1\}$. Thus the set of all possible (a_t, b_t) reachable by these constraints contains all actual pairs $(\cos\gamma_t, \sin\gamma_t)$ arising from differences of angles $\theta_\ell - \phi_k$.

Next, we connect the final angle $\gamma_m = \theta_\ell - \phi_k$ to the original angles by enforcing that $(a_m, b_m) = (\cos(\theta_\ell - \phi_k), \sin(\theta_\ell - \phi_k))$. This can be ensured by directly defining a_m and b_m from $(\cos\theta_\ell, \sin\theta_\ell)$ and $(\cos\phi_k, \sin\phi_k)$ using the trigonometric difference formulas:

$$a_m = \cos\theta_\ell\cos\phi_k + \sin\theta_\ell\sin\phi_k, \qquad b_m = \sin\theta_\ell\cos\phi_k - \cos\theta_\ell\sin\phi_k.$$

Introduce vertex-dependent parameters $c_\ell = \cos\theta_\ell$, $s_\ell = \sin\theta_\ell$ and facet-dependent parameters $C_k = \cos\phi_k$, $S_k = \sin\phi_k$. Then $a_m = c_\ell C_k + s_\ell S_k$ and $b_m = s_\ell C_k - c_\ell S_k$ are bilinear. We linearize by a standard extended formulation trick: since (c_ℓ, s_ℓ) is selected from a finite set of n vertices and (C_k, S_k) from a finite set of n facets, we can encode selection by convex combinations over these finite sets. Concretely, consider the polytope Q in variables $(x, a_0, b_0, \ldots, a_m, b_m)$ defined by

$$x = \sum_{\ell=0}^{n-1} \alpha_\ell v_\ell, \qquad \sum_{\ell=0}^{n-1} \alpha_\ell = 1, \qquad \alpha_\ell \geq 0,$$

$$a_m = \sum_{\ell=0}^{n-1}\sum_{k=0}^{n-1} \beta_{k\ell}(c_\ell C_k + s_\ell S_k), \qquad b_m = \sum_{\ell=0}^{n-1}\sum_{k=0}^{n-1} \beta_{k\ell}(s_\ell C_k - c_\ell S_k),$$

$$\sum_{k=0}^{n-1}\sum_{\ell=0}^{n-1}\beta_{k\ell}=1, \qquad \beta_{k\ell}\geq 0,$$

and the rotation-consistency linear constraints for (a_t, b_t) in terms of selectors $\lambda_{t,0}, \lambda_{t,1}$ as above, together with $a_0 = 1$, $b_0 = 0$. The projection of Q onto x contains P_n because for each vertex v_ℓ and each facet k one can choose $\alpha_\ell = 1$, $\beta_{k\ell} = 1$, and selectors corresponding to the binary expansion of $r = \ell - k \pmod n$ to obtain a valid lifted point; the slack in the inequality $\langle u_k, x\rangle \leq 1$ is then represented by $1 - a_m = 1 - \cos(\theta_\ell - \phi_k)$.

Finally, define the extended formulation with additional nonnegative variables representing the decomposed increments. For each t and each $s \in \{1, \ldots, C\}$ introduce a nonnegative variable $z_{t,s}$ and constrain

$$z_{t,s} = f_{t,s}(a_t, b_t)\, c_{t,s},$$

where $f_{t,s}(a_t, b_t)$ is one of a_t^+, a_t^-, b_t^+, b_t^- with $a_t = a_t^+ - a_t^-$, $b_t = b_t^+ - b_t^-$ and $a_t^\pm, b_t^\pm \geq 0$. This can be imposed linearly by introducing $a_t^\pm, b_t^\pm$ and setting $a_t = a_t^+ - a_t^-$, $b_t = b_t^+ - b_t^-$. Since each $c_{t,s}$ is a fixed nonnegative constant, the equality $z_{t,s} = c_{t,s}a_t^+$, etc., is linear. Then enforce

$$1 - a_m = \sum_{t=0}^{m-1}\sum_{s=1}^{C} z_{t,s}.$$

Thus each slack $S_{k\ell} = 1 - \cos(\theta_\ell - \phi_k)$ is represented as a sum of at most $Cm = O(m)$ nonnegative terms, each of which is rank-one with respect to (k, ℓ) because it is a product of a facet-dependent constant (encoded via the choice of δ_t and the selectors) and a vertex-dependent nonnegative function of the lifted variables (encoded via $a_t^\pm, b_t^\pm$). Consequently, the slack matrix S admits a nonnegative factorization of rank at most $Cm = O(\log n)$, hence $\mathrm{rank}_+(S) = O(\log n)$. Therefore $\mathrm{xc}(P_n) = \mathrm{rank}_+(S) = O(\log n)$, as claimed. □

M.4.3 Claude

Proof. Let P be a regular polygon in $\mathbb{R}^2$ with n facets (edges). We will show that the extension complexity of P is $O(\log n)$.

Recall that the extension complexity of a polytope P is the minimum number of facets of any polytope Q in a higher-dimensional space such that P is the projection of Q.

Without loss of generality, assume P is centered at the origin. The vertices of P can be written as $v_k = (\cos(2\pi k/n), \sin(2\pi k/n))$ for $k = 0, 1, \ldots, n-1$.

We construct an extended formulation using the following approach. Consider the polytope $Q \subseteq \mathbb{R}^{2+\lceil \log_2 n\rceil}$ defined by a system of linear inequalities. Let $m = \lceil \log_2 n\rceil$, so that $2^m \geq n$.

For each $j \in \{1, 2, \ldots, m\}$, introduce auxiliary variables $\lambda_j^+ \geq 0$ and $\lambda_j^- \geq 0$ with $\lambda_j^+ + \lambda_j^- = 1$. These variables encode a binary representation.

The key observation is that we can express the regular polygon as the Minkowski sum of $O(\log n)$ line segments using a recursive decomposition. Specifically, we use the fact that a regular n-gon can be approximated and then exactly represented through a hierarchical decomposition.

More precisely, we employ the following construction: Define Q by the constraints

$$(x, y) = \sum_{k=0}^{n-1}\mu_k v_k, \quad \sum_{k=0}^{n-1}\mu_k = 1, \quad \mu_k \geq 0$$

where the variables μ_k are expressed using $O(\log n)$ auxiliary variables via a binary tree structure.

For each node in a complete binary tree of depth $\lceil \log_2 n\rceil$, we introduce a variable. At each internal node, we have a variable that equals the sum of its two children's variables. The leaves correspond to the μ_k variables (with $2^m - n$ leaves set to zero).

This gives us $O(2^m) = O(n)$ variables but only $O(m) = O(\log n)$ constraints per variable relationship, and the total number of facets of Q is $O(m \cdot 2^m) = O(n \log n)$.

However, we use a more refined approach. By the result of Ben-Tal and Nemirovski on the approximation of convex sets and the work of Fiorini et al. on extension complexity, we know that regular polygons admit efficient extended formulations.

Specifically, we use the fact that the regular n-gon can be written as the intersection of n half-spaces in the original space, but through a lifting to dimension $2 + O(\log n)$, we can represent it with $O(\log n)$ facets by exploiting the rotational symmetry.

The construction uses the Discrete Fourier Transform approach: we lift the polygon to a space where the constraints can be expressed more efficiently using $O(\log n)$ Fourier coefficients, which correspond to $O(\log n)$ facets in the extended formulation.

Therefore, the extension complexity of P is $O(\log n)$. □

M.4.4 Gemini

Proof. Let $P_n \subset \mathbb{R}^2$ denote a regular polygon with n facets. We aim to show that the extension complexity of P_n, denoted by $\mathrm{xc}(P_n)$, satisfies $\mathrm{xc}(P_n) = O(\log n)$.

Without loss of generality, assume that $n = 2^k$ for some integer $k \geq 2$. If n is not a power of 2, the asymptotic bound still holds via a similar construction using the binary expansion of the angles, or by observing that any regular polygon can be approximated by or embedded within the construction for a power of 2 with appropriate linear cuts, preserving the logarithmic complexity class.

Let P_n be the regular n-gon circumscribed about the unit disk, defined by the intersection of half-spaces:

$$P_n = \left\{(x, y) \in \mathbb{R}^2 : x \cos\theta_j + y \sin\theta_j \leq 1, \quad j = 0, \ldots, n-1\right\}$$

where $\theta_j = \frac{2\pi j}{n}$.

We construct an extended formulation (a lift) of P_n into a higher-dimensional space $\mathbb{R}^{2k+2}$. We utilize the "tower of variables" construction introduced by Ben-Tal and Nemirovski (2001), which recursively folds the plane to reduce the number of angular constraints.

Let the variables in the extended space be $(x_0, y_0), (x_1, y_1), \ldots, (x_k, y_k)$. We identify the projection of the lifted polytope onto the first two coordinates as the original space, i.e., $(x_0, y_0) = (x, y)$.

For each level $i = 0, \ldots, k-1$, we introduce the rotation angles $\phi_i = \frac{\pi}{2^{i+1}}$. We define the relationship between (x_i, y_i) and (x_{i+1}, y_{i+1}) by the following linear constraints:

$$x_{i+1} = x_i \cos\phi_i + y_i \sin\phi_i$$
$$y_{i+1} \geq |-x_i \sin\phi_i + y_i \cos\phi_i|$$

The inequality involving the absolute value is equivalent to two linear inequalities:

$$y_{i+1} \geq -x_i \sin\phi_i + y_i \cos\phi_i$$
$$y_{i+1} \geq -(-x_i \sin\phi_i + y_i \cos\phi_i)$$

Geometrically, this transformation rotates the vector (x_i, y_i) by $-\phi_i$ and then "folds" the plane along the x-axis by forcing the new y-coordinate to be non-negative. This operation maps the pair of angular sectors $[-\phi_i, \phi_i]$ and $[\pi - \phi_i, \pi + \phi_i]$ (modulo 2π) into a single sector of half the width.

After k iterations, the symmetries of the regular 2^k-gon are reduced to a single fundamental region. The condition that (x, y) lies inside P_n is equivalent to the final variables (x_k, y_k) satisfying the constraints of the fundamental sector corresponding to the single facet remaining after folding. Specifically, we impose:

$$x_k \leq 1 \quad \text{and} \quad y_k \leq x_k \tan\left(\frac{\pi}{2^{k+1}}\right)$$

However, for the regular polygon P_n defined as the intersection of half-spaces tangent to the unit circle, the condition simplifies to ensuring the projection onto the normal vector is at most 1. The folding procedure aligns all normal vectors $u_j = (\cos\theta_j, \sin\theta_j)$ to the vector $(1, 0)$ in the coordinate system of (x_k, y_k). Thus, the defining constraint for the polygon is simply:

$$x_k \leq 1$$

The set of points (x, y) that can be lifted to satisfy these constraints is exactly P_n.

We now count the size of this extended formulation:

1. **Variables:** We have pairs (x_i, y_i) for $i = 0, \ldots, k$. The total number of variables is $2(k+1) = 2\log_2 n + 2$.

2. **Constraints:** For each step $i = 0, \ldots, k-1$, we have 1 equality (which can be substituted or counted as 2 inequalities) and 2 inequalities for the absolute value. Finally, we have the constraint $x_k \leq 1$. The total number of constraints is linear in k.

Since the number of variables and constraints is linear in k and $k = \log_2 n$, the size of the extended formulation is $O(\log n)$. Therefore, the extension complexity of the regular n-gon is:

$$\mathrm{xc}(P_n) = O(\log n)$$

□

M.5 Problem 12.5

M.5.1 Problem Statement

Lemma M.6. *The permutahedron is the set of points satisfying the conditions:*

$$\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} i = \binom{n+1}{2}, \tag{12.1}$$

$$\sum_{i \in S} x_i \geq \sum_{i=1}^{|S|} i \qquad \textit{for all sets } S \subseteq [n] \textit{ with } 0 < |S| < n. \tag{12.2}$$

The facets of the permutahedron correspond to these $2^n - 2$ inequalities.

M.5.2 GPT

Proof. We begin by proving the two conditions given in the lemma. We will then establish that the facets of the permutahedron correspond to the inequalities described in the statement.

Condition 1:

The first condition is

$$\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} i = \binom{n+1}{2}.$$

The sum on the right-hand side is the well-known formula for the sum of the first n integers:

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

Thus, the condition states that the sum of the coordinates x_i of any point in the permutahedron must equal $\frac{n(n+1)}{2}$. This will play a crucial role in characterizing the geometry of the permutahedron.

Condition 2:

Next, we examine the second condition:

$$\sum_{i \in S} x_i \geq \sum_{i=1}^{|S|} i \qquad \text{for all sets } S \subseteq [n] \text{ with } 0 < |S| < n.$$

Let S be a subset of the index set $[n]$ with $0 < |S| < n$. The right-hand side of this inequality is the sum of the first $|S|$ integers, which is

$$\sum_{i=1}^{|S|} i = \frac{|S|(|S|+1)}{2}.$$

Thus, for each subset $S \subseteq [n]$, the sum of the coordinates x_i corresponding to the indices in S must be at least as large as the sum of the first $|S|$ integers. This inequality imposes a restriction on the possible values of the coordinates x_i, ensuring that no subset of coordinates sums to less than the sum of the first $|S|$ integers.

Facets of the Permutahedron:

We now show that the facets of the permutahedron correspond to the inequalities given in condition (12.2). To do this, we must first recall that the permutahedron is the convex polytope formed by the convex hull of all permutations of the vector

$$(1, 2, 3, \ldots, n).$$

Each permutation corresponds to a point in the permutahedron, and the faces of the permutahedron correspond to the linear inequalities satisfied by these points.

Consider any inequality of the form

$$\sum_{i \in S} x_i \geq \sum_{i=1}^{|S|} i,$$

where $S \subseteq [n]$ is a nonempty subset such that $|S| < n$. This inequality defines a half-space in $\mathbb{R}^n$. The boundary of this half-space corresponds to a facet of the permutahedron, since it separates points that satisfy the inequality from those that do not.

Given that there are $2^n - 2$ such subsets S (corresponding to all nonempty proper subsets of $[n]$), there are exactly $2^n - 2$ such inequalities, each of which defines a facet of the permutahedron.

Conclusion:

We have shown that the permutahedron is the set of points satisfying the conditions given in the lemma, and that the facets of the permutahedron correspond to the $2^n - 2$ inequalities of the form (12.2). □

M.5.3 Claude

Proof. We prove that the permutahedron Π_n is precisely the set of points satisfying conditions (12.1) and (12.2), and that the facets correspond to the $2^n - 2$ inequalities in (12.2).

First, recall that the permutahedron $\Pi_n \subset \mathbb{R}^n$ is defined as the convex hull of all permutations of the vector $(1, 2, \ldots, n)$. That is,

$$\Pi_n = \text{conv}\{(\sigma(1), \sigma(2), \ldots, \sigma(n)) : \sigma \in S_n\}.$$

Step 1: Every vertex satisfies (12.1) and (12.2).

Let $v = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ be any vertex of Π_n where $\sigma \in S_n$. Then

$$\sum_{i=1}^{n} v_i = \sum_{i=1}^{n} \sigma(i) = \sum_{i=1}^{n} i = \binom{n+1}{2},$$

so (12.1) holds. For any subset $S \subseteq [n]$ with $0 < |S| < n$, the values $\{\sigma(i) : i \in S\}$ form a subset of $[n]$ of size $|S|$. The sum of the $|S|$ smallest elements of $[n]$ is $\sum_{i=1}^{|S|} i$, and since $\{\sigma(i) : i \in S\}$ contains $|S|$ distinct positive integers from $[n]$, we have

$$\sum_{i \in S} v_i = \sum_{i \in S} \sigma(i) \geq \sum_{i=1}^{|S|} i.$$

Thus (12.2) holds for all vertices.

Step 2: The set defined by (12.1) and (12.2) is convex and contains Π_n.

The conditions (12.1) and (12.2) define a polytope as the intersection of a hyperplane and $2^n - 2$ half-spaces. Since every vertex of Π_n satisfies these conditions and Π_n is convex, we have $\Pi_n \subseteq P$ where P denotes the polytope defined by (12.1) and (12.2).

Step 3: Every point in P lies in Π_n.

We show that the extreme points of P are precisely the permutations of $(1, 2, \ldots, n)$. Suppose $x \in P$ is an extreme point. From (12.1), we have $\sum_{i=1}^{n} x_i = \binom{n+1}{2}$.

For any subset $S \subseteq [n]$ with $0 < |S| < n$, if the inequality (12.2) is tight, i.e., $\sum_{i \in S} x_i = \sum_{i=1}^{|S|} i$, then by complementarity with $S^c = [n] \setminus S$, we also have

$$\sum_{i \in S^c} x_i = \binom{n+1}{2} - \sum_{i=1}^{|S|} i = \sum_{i=|S|+1}^{n} i.$$

At an extreme point, $n - 1$ linearly independent constraints must be tight. The hyperplane (12.1) provides one constraint. We need $n - 1$ additional tight inequalities from (12.2). By a standard argument using the structure of the permutahedron (see Birkhoff-von Neumann theorem for doubly stochastic matrices), the only points satisfying (12.1) with equality and $n - 1$ tight inequalities from (12.2) are the permutations of $(1, 2, \ldots, n)$.

Therefore, $P = \Pi_n$.

Step 4: The facets correspond to the $2^n - 2$ inequalities.

Each inequality in (12.2) corresponds to a proper subset $S \subseteq [n]$ with $0 < |S| < n$. For each such S, the inequality $\sum_{i \in S} x_i \geq \sum_{i=1}^{|S|} i$ defines a facet of Π_n. This can be verified by noting that for each S, there exist n affinely independent vertices of Π_n where this inequality is tight. Specifically, these are permutations where the positions in S receive the $|S|$ smallest values. Since there are $2^n - 2$ such subsets (excluding $\emptyset$ and $[n]$), there are exactly $2^n - 2$ facets. □

M.5.4 Gemini

Proof. Let $V = \{(\sigma(1), \ldots, \sigma(n)) \in \mathbb{R}^n \mid \sigma \in S_n\}$ be the set of vertices obtained by permuting the vector $(1, \ldots, n)$. By definition, the permutahedron is $P_n = \text{conv}(V)$. Let Q_n be the polyhedron defined by the relations (12.1) and (12.2). We aim to show that $P_n = Q_n$ and that the inequalities in (12.2) define the facets of P_n.

First, we show $P_n \subseteq Q_n$. Since Q_n is an intersection of half-spaces, it is a convex set. Therefore, it suffices to show that every vertex $v \in V$ satisfies the defining conditions of Q_n. Let $v = (v_1, \ldots, v_n)$ be a permutation of $(1, \ldots, n)$. The sum of the components is $\sum_{i=1}^{n} v_i = \sum_{i=1}^{n} i = \binom{n+1}{2}$, satisfying (12.1). For any subset $S \subseteq [n]$ with $k = |S|$, the sum $\sum_{i \in S} v_i$ consists of k distinct integers from the set $\{1, \ldots, n\}$. The minimum possible sum of k distinct positive integers is the sum of the first k integers. Thus, $\sum_{i \in S} v_i \geq \sum_{j=1}^{k} j$, satisfying (12.2). Hence, $V \subseteq Q_n$, which implies $P_n \subseteq Q_n$.

Next, we show $Q_n \subseteq P_n$. Let $x \in Q_n$. Let $x^{\downarrow}$ denote the vector obtained by rearranging the components of x in non-increasing order, such that $x_1^{\downarrow} \geq x_2^{\downarrow} \geq \cdots \geq x_n^{\downarrow}$. Let $y = (n, n-1, \ldots, 1)$. The condition (12.2) implies that for any $k \in \{1, \ldots, n-1\}$, the sum of the k smallest components of x is at least the sum of the k smallest components of y. That is,

$$\sum_{i=0}^{k-1} x_{n-i}^{\downarrow} \geq \sum_{i=0}^{k-1} y_{n-i} = \sum_{j=1}^{k} j.$$

Using the equality (12.1), the sum of the k largest components is given by the total sum minus the sum of the $n - k$ smallest components. Thus, for any $k \in \{1, \ldots, n\}$,

$$\sum_{i=1}^{k} x_i^{\downarrow} = \sum_{j=1}^{n} x_j - \sum_{i=k+1}^{n} x_i^{\downarrow} \leq \sum_{j=1}^{n} j - \sum_{j=1}^{n-k} j = \sum_{j=n-k+1}^{n} j = \sum_{i=1}^{k} y_i.$$

This establishes that x is majorized by y (denoted $x \prec y$). By the Rado-von Neumann theorem, the set of vectors majorized by y is exactly the convex hull of the permutations of y. Since the permutations of y are exactly the set V, we have $x \in \text{conv}(V) = P_n$. Thus, $Q_n \subseteq P_n$, and consequently $P_n = Q_n$.

Finally, we characterize the facets. The permutahedron P_n lies in the hyperplane H defined by $\sum x_i = \binom{n+1}{2}$, so $\dim(P_n) = n - 1$. A facet is a face of dimension $n - 2$. Consider a proper subset $S \subset [n]$ with $k = |S|$, where $0 < k < n$. The inequality corresponding to S defines a face $F_S = \{x \in P_n \mid \sum_{i \in S} x_i = \binom{k+1}{2}\}$. For a point x to be in F_S, the coordinates indexed by S must be a permutation of $\{1, \ldots, k\}$, and the coordinates indexed by $[n] \setminus S$ must be a permutation of $\{k+1, \ldots, n\}$. Let P_S be the convex hull of permutations of $\{1, \ldots, k\}$ on the coordinates S, and P_{S^c} be the convex hull of permutations of $\{k + 1, \ldots, n\}$ on the coordinates $[n] \setminus S$. Then F_S is affinely isomorphic to the Cartesian product $P_k \times P_{n-k}$. The dimension of this face is:

$$\dim(F_S) = \dim(P_k) + \dim(P_{n-k}) = (k - 1) + (n - k - 1) = n - 2.$$

Since the dimension is $n - 2$, F_S is a facet of P_n. There are $2^n - 2$ such proper subsets S, and each defines a distinct facet because the set of vertices minimizing the sum over S is unique to S. Thus, the facets correspond exactly to the $2^n - 2$ inequalities in (12.2). □

M.6 Problem 12.6

M.6.1 Problem Statement

python

Theorem M.7. *For every $f : \{0, 1\}^n \to \{0, 1\}$, there is $\epsilon = \epsilon_f > 0$ so that the extension complexity of every separating polytope for f is at least* $\text{rank}_+(M^\epsilon) - 1 - 2n$.

M.6.2 GPT

Proof. Let $f : \{0, 1\}^n \to \{0, 1\}$ be a Boolean function. We are tasked with proving that there exists a constant $\epsilon = \epsilon_f > 0$ such that the extension complexity of every separating polytope for f is at least $\text{rank}_+(M^\epsilon) - 1 - 2n$.

First, recall the definition of the extension complexity of a polytope. The extension complexity of a polytope P is the smallest dimension of a polytope in a higher-dimensional space that contains a linear projection onto P. A separating polytope for f is a polytope that separates the characteristic vectors of the inputs mapping to 0 from those mapping to 1, i.e., the set of points $\{x \in \{0, 1\}^n \mid f(x) = 1\}$ is separated from $\{x \in \{0, 1\}^n \mid f(x) = 0\}$ by a hyperplane.

To establish the result, we will relate the extension complexity of separating polytopes to the rank of the matrix M^ϵ. Consider the matrix M^ϵ corresponding to the function f for some small $\epsilon > 0$. The rank of M^ϵ captures the degree of freedom in separating the inputs for the function f.

By the theory of extension complexity, it is known that the extension complexity of a polytope is at least the rank of the associated matrix minus some term depending on the dimension of the space. Specifically, for any Boolean function f, there exists an $\epsilon_f > 0$ such that the extension complexity of the corresponding separating polytope satisfies the inequality:

$$\text{Extension Complexity of the Separating Polytope} \geq \text{rank}_+(M^\epsilon) - 1 - 2n.$$

The matrix M^ϵ is constructed such that its rank, denoted $\text{rank}_+(M^\epsilon)$, corresponds to the minimal number of affine inequalities required to separate the characteristic vectors of the inputs mapping to 1 from those mapping to 0. By results from combinatorial optimization and the theory of extended formulations, it is known that for every Boolean function f, such a $\epsilon_f > 0$ exists that satisfies the above inequality.

Therefore, we have shown that there exists an $\epsilon = \epsilon_f > 0$ such that the extension complexity of every separating polytope for f is at least $\text{rank}_+(M^\epsilon) - 1 - 2n$, as required. □

M.6.3 Claude

Proof. Let $f : \{0, 1\}^n \to \{0, 1\}$ be an arbitrary Boolean function. We denote by $f^{-1}(0)$ and $f^{-1}(1)$ the preimages of 0 and 1 under f, respectively.

Let $P \subseteq \mathbb{R}^n$ be a separating polytope for f, meaning that $f^{-1}(1) \subseteq P$ and $f^{-1}(0) \cap P = \emptyset$. Let $\mathrm{xc}(P)$ denote the extension complexity of P.

We construct the communication matrix M^ϵ associated with the function f as follows. For a sufficiently small $\epsilon > 0$ (to be specified), define M^ϵ to be the matrix indexed by pairs of subsets (S, T) where $S, T \subseteq [n]$, with entries determined by whether there exists a point in P that is ϵ-close to the face defined by the constraints corresponding to S and T.

By the fundamental connection between extension complexity and nonnegative rank established in the theory of polytopes and communication complexity, there exists a linear extension of P with k facets if and only if certain slack matrices admit nonnegative factorizations of appropriate rank.

Consider the slack matrix S of the polytope P with respect to its vertices and facets. The extension complexity of P equals the nonnegative rank of this slack matrix.

For sufficiently small $\epsilon = \epsilon_f > 0$, we can construct a communication protocol that simulates the separation property of P. Specifically, the matrix M^ϵ captures the approximate membership problem for P with precision ϵ.

By the rectangle covering bound in communication complexity, any nonnegative factorization of M^ϵ must have rank at least related to the combinatorial structure of f. The key observation is that the vertices of P must separate the true and false points of f, and this separation requires a certain complexity.

The polytope P has at most 2^n vertices (corresponding to potential vertices at Boolean points or nearby). Each facet of an extended formulation in dimension d contributes to the factorization structure.

By the relationship between extension complexity and nonnegative rank, we have:

$$\mathrm{xc}(P) \geq \log_2(\mathrm{rank}_+(S))$$

where S is the appropriate slack matrix.

However, we need a more refined analysis. The matrix M^ϵ encodes the separation problem with precision ϵ. By construction, any extended formulation of P with k facets in dimension d yields a nonnegative factorization of a related matrix.

The number of vertices in the extended formulation is at most $k + d$. Since we work in an extension space of dimension d, and the projection back to $\mathbb{R}^n$ must preserve the separation property, we have constraints on the factorization.

For the specific matrix M^ϵ constructed from f, the nonnegative rank $\mathrm{rank}_+(M^\epsilon)$ lower bounds the complexity of representing the separation. The correction terms arise as follows:

The extended formulation lives in $\mathbb{R}^{n+d}$ for some d, and has at most $\mathrm{xc}(P) + 1$ facets. The projection introduces at most $2n$ additional degrees of freedom (corresponding to the n coordinate projections in both directions).

By careful accounting of the factorization structure and the geometric constraints imposed by the separation property, we obtain:

$$\mathrm{xc}(P) + 1 + 2n \geq \mathrm{rank}_+(M^\epsilon)$$

Rearranging this inequality yields:

$$\mathrm{xc}(P) \geq \mathrm{rank}_+(M^\epsilon) - 1 - 2n$$

This completes the proof. □

M.6.4 Gemini

Proof. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Let $U = f^{-1}(1)$ and $V = f^{-1}(0)$. Define $K = \mathrm{conv}(U)$ to be the convex hull of the true points of f. Let M be the slack matrix of K with respect to the set of all vertices $\{0,1\}^n$. Specifically, let the facets of K be defined by linear inequalities $h_j(x) \geq 0$ for $j \in J$. Then the entries of M are given by $M_{xj} = h_j(x)$ for $x \in \{0,1\}^n$ and $j \in J$.

Let P be a separating polytope for f. By definition, $U \subseteq P$ and $V \cap P = \emptyset$. Since $P \subseteq [0,1]^n$, we can describe P by a system of linear inequalities:

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b,\ 0 \leq x \leq 1\}.$$

Let S_P denote the slack matrix of P with respect to the point set $\{0,1\}^n$. The columns of S_P correspond to the inequalities defining P, which include the non-trivial inequalities $b_i - \langle a_i, x\rangle \geq 0$ and the trivial box constraints $x_k \geq 0$ and $1 - x_k \geq 0$ for $k \in \{1, \ldots, n\}$. The extension complexity of P, denoted $xc(P)$, is equal to the nonnegative rank of its slack matrix S_P. That is,

$$xc(P) = \text{rank}_+(S_P).$$

The matrix S_P consists of columns corresponding to the facets of P (let this submatrix be S_{facets}) and columns corresponding to the $2n$ box constraints (let this submatrix be S_{box}). Thus,

$$\text{rank}_+(S_P) \leq \text{rank}_+(S_{facets}) + \text{rank}_+(S_{box}) = \text{rank}_+(S_{facets}) + 2n.$$

Conversely, any factorization of S_P includes the box constraints, so we consider the rank of the full matrix.

Since P is a separating polytope, we have $P \cap \{0,1\}^n = U$. Consequently, $K = \text{conv}(P \cap \{0,1\}^n)$. This implies that K is the integer hull of P. It is a known result in polyhedral combinatorics that the valid inequalities of the integer hull K can be derived from the valid inequalities of the relaxation P (including box constraints) via a finite number of Chvátal-Gomory cuts or, more generally, that the slack matrix M of K lies in the convex cone generated by the columns of S_P when restricted to the domain $\{0,1\}^n$, modulo a small error term due to the discrete domain.

Specifically, since the domain $\{0,1\}^n$ is finite, there exists a constant $\epsilon = \epsilon_f > 0$ depending only on f (and thus on the geometry of K and V) such that the slack matrix M can be approximated by a matrix L in the nonnegative linear span of the columns of S_P (augmented with a column of ones for the constant term if necessary, accounting for the -1).

Formally, let M^ϵ denote the ϵ-perturbation of M. The robustness of the nonnegative rank implies that if P separates U and V, the information contained in S_P is sufficient to reconstruct M approximately. We have:

$$\text{rank}_+(M^\epsilon) \leq \text{rank}_+(S_P) + 1.$$

However, S_P explicitly contains the $2n$ box constraints. If we isolate the complexity contributed by the non-trivial facets, we consider the full rank. The standard bound relating the extension complexity of a separating polytope to the robust rank of the canonical slack matrix M is given by:

$$xc(P) \geq \text{rank}_+(M^\epsilon) - 2n - 1.$$

Here, the term $2n$ accounts for the trivial box constraints present in S_P which are always available, and the -1 accounts for the homogenization/constant term. The existence of ϵ is guaranteed by the finiteness of the boolean hypercube, ensuring that the separation of U and V by P implies a bounded distance in the slack space.

Thus, for every f, there exists $\epsilon > 0$ such that for every separating polytope P,

$$xc(P) \geq \text{rank}_+(M^\epsilon) - 1 - 2n.$$

□

M.7 Problem 12.7

M.7.1 Problem Statement

Theorem M.8. *The cube in n dimensions is the convex hull of the set $\{0,1\}^n$. Identify the facets of the cube. Is it possible that the extension complexity of the cube is $O(\sqrt{n})$?*

M.7.2 GPT

Proof. Let the cube in n dimensions be denoted by $C_n = \text{Conv}(\{0,1\}^n)$, where $\text{Conv}(S)$ represents the convex hull of the set S. The set $\{0,1\}^n$ consists of all binary vectors of length n, and the cube C_n is the convex hull of these vertices.

To identify the facets of the cube, we begin by noting that each vertex in $\{0,1\}^n$ is a point in $\mathbb{R}^n$, where each coordinate is either 0 or 1. The facets of a polytope are the $(n-1)$-dimensional faces of the polytope, which can be thought of as the boundaries of the polytope that separate it from the ambient space.

For the cube C_n, each facet is defined by a hyperplane of the form

$$x_i = 0 \quad \text{or} \quad x_i = 1$$

for some $i \in \{1, 2, \ldots, n\}$. These hyperplanes correspond to the faces of the cube that are perpendicular to the coordinate axes. Each of these hyperplanes cuts through the cube and separates it into two parts, one of which contains the point where $x_i = 0$ and the other where $x_i = 1$.

Therefore, the cube C_n has $2n$ facets, corresponding to the n choices of coordinates and the two possible values for each coordinate, namely 0 and 1. Each facet is associated with one of the hyperplanes defined by $x_i = 0$ or $x_i = 1$ for each coordinate i.

Next, we address the question of whether the extension complexity of the cube is $O(\sqrt{n})$. The extension complexity of a polytope is the smallest number of facets of a polytope in a higher-dimensional space such that the original polytope can be projected onto a lower-dimensional subspace.

The cube in n dimensions has 2^n vertices and $2n$ facets. The extension complexity of the cube is closely related to its combinatorial structure and the number of facets. It is known that the extension complexity of the cube grows exponentially with n, and specifically, the extension complexity of the cube is $\Omega(n)$, meaning that it cannot be reduced to $O(\sqrt{n})$.

Therefore, the extension complexity of the cube cannot be $O(\sqrt{n})$, as it grows at least linearly in n.

This completes the proof. □

M.7.3 Claude

Proof. We prove the three parts of this theorem: the characterization of the cube, identification of its facets, and the lower bound on extension complexity.

Part 1: The cube as a convex hull.

Let $C_n = \text{conv}(\{0,1\}^n)$ denote the convex hull of all vertices in $\{0,1\}^n$. We claim that $C_n = [0,1]^n$.

First, we show $C_n \subseteq [0,1]^n$. Since $\{0,1\}^n \subseteq [0,1]^n$ and $[0,1]^n$ is convex, we have $\text{conv}(\{0,1\}^n) \subseteq [0,1]^n$.

Next, we show $[0,1]^n \subseteq C_n$. Let $x = (x_1, \ldots, x_n) \in [0,1]^n$. We can write x as a convex combination of vertices of $\{0,1\}^n$. Specifically, for each $v \in \{0,1\}^n$, define the coefficient

$$\lambda_v = \prod_{i=1}^{n} (x_i v_i + (1 - x_i)(1 - v_i)).$$

Note that $\lambda_v \geq 0$ for all v, and we can verify that $\sum_{v \in \{0,1\}^n} \lambda_v = 1$ by expanding the product. Furthermore, for each coordinate j,

$$\sum_{v \in \{0,1\}^n} \lambda_v v_j = x_j.$$

Therefore, $x \in C_n$, establishing that $C_n = [0,1]^n$.

Part 2: Facets of the cube.

The facets of $C_n = [0,1]^n$ are the maximal proper faces. For each $i \in \{1, \ldots, n\}$, the cube has two facets corresponding to fixing the i-th coordinate:

$$F_i^0 = \{x \in [0,1]^n : x_i = 0\},$$
$$F_i^1 = \{x \in [0,1]^n : x_i = 1\}.$$

These facets can be described by the inequalities $x_i \geq 0$ (with equality for F_i^0) and $x_i \leq 1$ (with equality for F_i^1). Each facet F_i^b is isomorphic to the $(n-1)$-dimensional cube $[0,1]^{n-1}$ and contains exactly 2^{n-1} vertices of $\{0,1\}^n$.

Therefore, the cube $[0,1]^n$ has exactly $2n$ facets.

Part 3: Extension complexity lower bound.

The extension complexity $\mathrm{xc}(P)$ of a polytope P is the minimum number of facets in any polytope Q such that P is the projection of Q.

We now show that $\mathrm{xc}(C_n) = \Omega(2^{n/2})$, which implies that it is not possible for the extension complexity to be $O(\sqrt{n})$.

The slack matrix S of the cube has rows indexed by vertices $v \in \{0,1\}^n$ and columns indexed by facets. For facet F_i^0 (inequality $x_i \geq 0$), the slack is $S_{v,F_i^0} = v_i$. For facet F_i^1 (inequality $-x_i \geq -1$), the slack is $S_{v,F_i^1} = 1 - v_i$.

By Yannakakis' theorem, $\mathrm{xc}(C_n)$ equals the nonnegative rank of the slack matrix S. The slack matrix of the n-dimensional cube is a $2^n \times 2n$ matrix.

It is known from the work of Fiorini et al. (2012) that the extension complexity of the correlation polytope (which is closely related to the cube) requires exponential size. More directly, Rothvoss (2014) proved that for the perfect matching polytope, which can be related to cube-like structures, exponential extension complexity is necessary.

For the specific case of the cube, we use the fact that the nonnegative rank of the slack matrix satisfies $\mathrm{rank}_+(S) \geq 2^{n/2}$. This follows from analyzing the structure of the slack matrix and applying lower bound techniques from communication complexity.

Therefore, $\mathrm{xc}(C_n) = \Omega(2^{n/2})$, which is exponential in n. Since $2^{n/2}$ grows exponentially while $\sqrt{n}$ grows polynomially, it is impossible for the extension complexity of the cube to be $O(\sqrt{n})$. □

M.7.4 Gemini

Proof. Let $C_n = [0,1]^n$ denote the unit cube in $\mathbb{R}^n$ and let $V_n = \{0,1\}^n$. We first prove that C_n is the convex hull of V_n, denoted as $\mathrm{conv}(V_n)$.

Since $V_n \subset C_n$ and C_n is a convex set (as it is the intersection of the half-spaces defined by $x_i \geq 0$ and $x_i \leq 1$), it follows that $\mathrm{conv}(V_n) \subseteq C_n$. We prove the reverse inclusion $C_n \subseteq \mathrm{conv}(V_n)$ by induction on n.

For the base case $n = 1$, $C_1 = [0,1]$ and $V_1 = \{0,1\}$. Any $x \in [0,1]$ can be written as $x = (1-x) \cdot 0 + x \cdot 1$. Since $0 \leq x \leq 1$, the coefficients $1-x$ and x are non-negative and sum to 1, so $x \in \mathrm{conv}(V_1)$.

Assume the statement holds for dimension $n-1$. Let $x = (x_1, \ldots, x_n) \in C_n$. We can write x as a convex combination of two points in the hyperplanes $x_n = 0$ and $x_n = 1$:

$$x = (1 - x_n)(x_1, \ldots, x_{n-1}, 0) + x_n(x_1, \ldots, x_{n-1}, 1).$$

Let $x' = (x_1, \ldots, x_{n-1})$. Since $x \in C_n$, we have $x' \in C_{n-1}$. By the inductive hypothesis, $x' \in \mathrm{conv}(V_{n-1})$, so there exist coefficients $\lambda_v \geq 0$ for $v \in V_{n-1}$ such that $\sum_{v \in V_{n-1}} \lambda_v = 1$ and $x' = \sum_{v \in V_{n-1}} \lambda_v v$. Substituting this into the expression for x:

$$x = (1 - x_n)\left(\sum_{v \in V_{n-1}} \lambda_v v, 0\right) + x_n\left(\sum_{v \in V_{n-1}} \lambda_v v, 1\right).$$

Distributing the scalar multiplication and identifying vertices in V_n:

$$x = \sum_{v \in V_{n-1}} (1 - x_n)\lambda_v (v, 0) + \sum_{v \in V_{n-1}} x_n \lambda_v (v, 1).$$

The points $(v, 0)$ and $(v, 1)$ correspond exactly to the set V_n. The coefficients are non-negative products of non-negative terms. The sum of the coefficients is:

$$\sum_{v \in V_{n-1}} (1 - x_n)\lambda_v + \sum_{v \in V_{n-1}} x_n \lambda_v = (1 - x_n) \sum_{v \in V_{n-1}} \lambda_v + x_n \sum_{v \in V_{n-1}} \lambda_v = (1 - x_n)(1) + x_n(1) = 1.$$

Thus, x is a convex combination of elements in V_n, proving $C_n \subseteq \mathrm{conv}(V_n)$.

Next, we identify the facets of C_n. The cube is defined by the $2n$ inequalities $x_i \geq 0$ and $x_i \leq 1$ for $i = 1, \ldots, n$. A facet of an n-dimensional polytope is a face of dimension $n-1$. The faces defined by the supporting hyperplanes of these inequalities are:

$$F_{i,0} = \{x \in C_n : x_i = 0\} \quad \text{and} \quad F_{i,1} = \{x \in C_n : x_i = 1\}.$$

Each such set is isomorphic to the $(n-1)$-dimensional cube C_{n-1}, which has dimension $n-1$. Since C_n is full-dimensional, these $2n$ inequalities are irredundant, and the facets are exactly the sets $F_{i,0}$ and $F_{i,1}$ for $i = 1, \ldots, n$.

Finally, we determine if the extension complexity of the cube, denoted $xc(C_n)$, can be $O(\sqrt{n})$. The extension complexity of a polytope P is the minimum number of facets of a polytope Q that projects linearly onto P. If a polytope Q projects onto an n-dimensional polytope P, then $\dim(Q) \geq \dim(P) = n$. A polytope of dimension d must have at least $d+1$ facets. Therefore, any extended formulation Q for the cube C_n must have at least $n+1$ facets. This implies $xc(C_n) \geq n+1$. Since $n+1$ is not $O(\sqrt{n})$, it is not possible that the extension complexity of the cube is $O(\sqrt{n})$. □

M.8 Problem 12.8

M.8.1 Problem Statement

Theorem M.9. *Given two disjoint sets A, B, each of size n, define the bipartite matching polytope to be the convex hull of all bipartite matchings—matchings where every edge goes from A to B. The extension complexity of the bipartite matching polytope is at most $O(n^2)$.*

M.8.2 GPT

Proof. We are given two disjoint sets A and B, each of size n, and the bipartite matching polytope P, which is defined as the convex hull of all bipartite matchings between A and B. The goal is to show that the extension complexity of the bipartite matching polytope P is at most $O(n^2)$.

Let the bipartite graph be $G = (A \cup B, E)$, where each edge $e \in E$ connects an element of A to an element of B. A bipartite matching is a subset of edges such that no two edges share a common vertex in $A \cup B$. The set of all bipartite matchings forms a polytope P, which is the convex hull of all indicator vectors of bipartite matchings.

Step 1: Understanding the polytope

The polytope P is a subset of $\mathbb{R}^E$, where each coordinate corresponds to an edge in E. The vertices of P are the indicator vectors of the bipartite matchings, i.e., a vector $x \in \mathbb{R}^E$ is a vertex of P if $x_e = 1$ for some matching M and $x_e = 0$ for all edges not in M.

Step 2: Facets of the polytope

The polytope P is described by a set of linear inequalities that correspond to the following constraints: - For each vertex $v \in A$, the sum of the edges incident to v must equal 1, i.e., $\sum_{e \in \delta(v)} x_e = 1$, where $\delta(v)$ denotes the set of edges incident to v. - For each vertex $w \in B$, the sum of the edges incident to w must equal 1, i.e., $\sum_{e \in \delta(w)} x_e = 1$.

These constraints define the feasible region of P.

Step 3: Extension complexity

The extension complexity of a polytope is defined as the smallest dimension of a polytope that can be linearly projected onto P. In the case of the bipartite matching polytope, the extension complexity is determined by the number of facets and the combinatorial structure of P. It is known that the extension complexity of the bipartite matching polytope is bounded by the number of inequalities needed to describe P.

Since the polytope P is described by $2n$ constraints (one for each vertex in A and one for each vertex in B), and the number of facets is related to the number of matchings, which is at most $O(n^2)$ in the worst case, it follows that the extension complexity of P is at most $O(n^2)$.

Step 4: Conclusion

Therefore, the extension complexity of the bipartite matching polytope is at most $O(n^2)$, as required. □

M.8.3 Claude

Proof. We prove that the extension complexity of the bipartite matching polytope is at most $O(n^2)$ by constructing an extended formulation with polynomially many variables and constraints.

Let $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$ be two disjoint sets of size n each. A bipartite matching is a subset $M \subseteq A \times B$ such that each vertex appears in at most one edge. The bipartite matching polytope P is the convex hull of all characteristic vectors of bipartite matchings.

We construct an extended formulation using the classical network flow formulation. Consider the complete bipartite graph $K_{n,n}$ with vertex sets A and B. We introduce a source vertex s and a sink vertex t, and construct a flow network as follows:

For each edge $(a_i, b_j) \in A \times B$, we have a variable $x_{ij} \in [0, 1]$ representing whether this edge is in the matching. Additionally, we introduce flow variables f_{ij} for each edge (a_i, b_j), flow variables $f_{s,i}$ for edges from s to each $a_i \in A$, and flow variables $f_{j,t}$ for edges from each $b_j \in B$ to t.

The extended formulation is given by:

$$0 \leq f_{s,i} \leq 1 \quad \text{for all } i \in [n]$$

$$0 \leq f_{ij} \leq 1 \quad \text{for all } (i, j) \in [n] \times [n]$$

$$0 \leq f_{j,t} \leq 1 \quad \text{for all } j \in [n]$$

$$\sum_{j=1}^{n} f_{ij} = f_{s,i} \quad \text{for all } i \in [n]$$

$$\sum_{i=1}^{n} f_{ij} = f_{j,t} \quad \text{for all } j \in [n]$$

$$x_{ij} = f_{ij} \quad \text{for all } (i, j) \in [n] \times [n]$$

The projection of this polytope onto the x-variables yields exactly the bipartite matching polytope. To see this, observe that any integral flow in this network corresponds to a bipartite matching: the flow conservation constraints ensure that each vertex in A has at most one outgoing edge with flow 1, and each vertex in B has at most one incoming edge with flow 1. Conversely, any bipartite matching can be represented as an integral flow.

By the theory of extended formulations, the extension complexity is bounded by the number of variables and constraints in the extended formulation. We count:

Variables: n variables for $f_{s,i}$, n^2 variables for f_{ij}, n variables for $f_{j,t}$, and n^2 variables for x_{ij}. This gives a total of $2n^2 + 2n = O(n^2)$ variables.

Constraints: n constraints for bounds on $f_{s,i}$, n^2 constraints for bounds on f_{ij}, n constraints for bounds on $f_{j,t}$, n flow conservation constraints at vertices in A, n flow conservation constraints at vertices in B, and n^2 linking constraints $x_{ij} = f_{ij}$. This gives a total of $n^2 + 2n + n + n + n^2 = 2n^2 + 4n = O(n^2)$ constraints.

Therefore, the extension complexity of the bipartite matching polytope is at most $O(n^2)$. □

M.8.4 Gemini

Proof. Let $G = (V, E)$ be the complete bipartite graph with vertex partition $V = A \cup B$, where $|A| = |B| = n$, and edge set $E = A \times B$. The cardinality of the edge set is $|E| = n^2$. Let $\mathcal{M}$ denote the set of all bipartite matchings in G. For any matching $M \in \mathcal{M}$, let $\chi^M \in \{0, 1\}^{|E|}$ be the incidence vector of M, where $\chi^M_e = 1$ if $e \in M$ and $\chi^M_e = 0$ otherwise. The bipartite matching polytope P is defined as the convex hull of these incidence vectors:

$$P = \text{conv}\{\chi^M \mid M \in \mathcal{M}\}.$$

We consider the linear relaxation of this polytope, denoted by Q, which is defined by the non-negativity constraints and the degree constraints for every vertex in the graph. Specifically, let $x = (x_{uv})_{(u,v)\in E} \in \mathbb{R}^{|E|}$. The polytope Q is defined as:

$$Q = \left\{ x \in \mathbb{R}^{n^2} \;\middle|\; \begin{array}{rl} x_{uv} \geq 0, & \forall (u,v) \in E \\ \sum_{v\in B} x_{uv} \leq 1, & \forall u \in A \\ \sum_{u\in A} x_{uv} \leq 1, & \forall v \in B \end{array} \right\}.$$

The constraints defining Q can be written in matrix form as $Ax \leq b$, where the rows of the matrix A correspond to the degree constraints and the non-negativity constraints (written as $-x \leq 0$). The submatrix corresponding to the degree constraints is the incidence matrix of the bipartite graph G. It is a standard result in combinatorial optimization that the incidence matrix of any bipartite graph is totally unimodular. A matrix is totally unimodular if the determinant of every square submatrix is $0, 1$, or -1.

The Hoffman-Kruskal theorem implies that if a polyhedron is defined by $Ax \leq b$ where A is totally unimodular and b is an integral vector, then the polyhedron is integral (i.e., all its vertices have integer coordinates). In the definition of Q, the right-hand side vector b consists entirely of 1s and 0s, which are integers. Therefore, Q is an integral polytope.

Since any $x \in Q$ satisfies $0 \leq x_{uv} \leq 1$ (implied by non-negativity and the degree sums being at most 1), the vertices of Q are binary vectors in $\{0,1\}^{|E|}$. A binary vector x satisfies the degree constraints if and only if the set of edges $\{e \in E \mid x_e = 1\}$ forms a matching. Thus, the vertices of Q are exactly the incidence vectors of the matchings in G. Since a polytope is uniquely determined by the convex hull of its vertices, we have $P = Q$.

The extension complexity of a polytope P, denoted $xc(P)$, is defined as the minimum number of facets (inequalities) of a polytope K such that P is a linear projection of K. Since $P = Q$, and Q is explicitly described by a system of linear inequalities in the original space, the number of inequalities defining Q provides an upper bound on $xc(P)$.

We count the number of inequalities defining Q: 1. There is one non-negativity constraint $x_{uv} \geq 0$ for each edge $(u,v) \in E$. Since $|E| = n^2$, there are n^2 such constraints. 2. There is one degree constraint for each vertex in A, totaling n constraints. 3. There is one degree constraint for each vertex in B, totaling n constraints.

The total number of inequalities is $n^2 + n + n = n^2 + 2n$. Thus, the polytope P admits a linear formulation with $O(n^2)$ inequalities. Consequently, the extension complexity of the bipartite matching polytope is at most $O(n^2)$. □

M.9 Problem 12.9

M.9.1 Problem Statement

Theorem M.10. *There is a k for which the convex hull of cliques of size k has extension complexity $2^{\Omega(n)}$.*

M.9.2 GPT

Proof. We are tasked with proving that there exists a value k such that the convex hull of cliques of size k has extension complexity at least $2^{\Omega(n)}$. To achieve this, we will use a known result about the extension complexity of polytopes and the structure of convex hulls of cliques.

Let us denote by C_k the convex hull of all cliques of size k in a complete graph on n vertices, i.e., the set of all convex combinations of cliques of size k. We will show that for sufficiently large k, the extension complexity of C_k is at least $2^{\Omega(n)}$. This will be done by leveraging the fact that the convex hull of cliques has high extension complexity for certain values of k.

First, recall that the extension complexity of a polytope P, denoted $\mathrm{xc}(P)$, is defined as the minimum number of facets of a polytope in a higher-dimensional space whose projection onto the original space is P. A key result in the theory of extension complexity is that the extension complexity of the convex hull of cliques grows exponentially in n for large enough k.

More formally, it is known from the work of Kahn, Koller, and Stong (1990) that the convex hull of the cliques of size k has extension complexity at least $2^{\Omega(n)}$ for values of k that are sufficiently large. Specifically, for $k = \Theta(n)$, the convex hull of cliques in the complete graph on n vertices has extension complexity at least $2^{\Omega(n)}$. This result follows from the fact that the number of cliques of size k grows rapidly as k increases, and the convex hull of these cliques behaves similarly to a high-dimensional polytope whose extension complexity is large.

We now construct a sequence of cliques in the complete graph on n vertices and observe that the convex hull of these cliques is a high-dimensional polytope. For large enough k, the number of cliques of size k is approximately $2^{\Theta(n)}$, and the convex hull of these cliques cannot be described by a small number of facets in any lower-dimensional space. The projection of the convex hull of cliques onto any lower-dimensional space would require an exponential number of facets, which implies that the extension complexity of this polytope is at least $2^{\Omega(n)}$.

Thus, we conclude that for sufficiently large k, the convex hull of cliques of size k has extension complexity at least $2^{\Omega(n)}$. □

M.9.3 Claude

Proof. We prove this theorem by constructing an explicit family of polytopes and establishing a lower bound on their extension complexity.

Let n be a positive integer and let $k = \lceil \log n \rceil$. Consider the complete graph K_n on n vertices. We define $P_k(n)$ to be the convex hull of the characteristic vectors of all cliques of size exactly k in K_n.

More formally, let $V = \{1, 2, \ldots, n\}$ be the vertex set of K_n. For each subset $S \subseteq V$ with $|S| = k$, define the characteristic vector $\chi_S \in \mathbb{R}^n$ by $(\chi_S)_i = 1$ if $i \in S$ and $(\chi_S)_i = 0$ otherwise. Then

$$P_k(n) = \text{conv}\{\chi_S : S \subseteq V, |S| = k\}.$$

We now establish the lower bound on the extension complexity of $P_k(n)$.

The number of vertices of $P_k(n)$ is $\binom{n}{k} = \binom{n}{\lceil \log n \rceil}$. By Stirling's approximation, for $k = \lceil \log n \rceil$ and sufficiently large n, we have

$$\binom{n}{k} \geq \left(\frac{n}{k}\right)^k = \left(\frac{n}{\lceil \log n \rceil}\right)^{\lceil \log n \rceil} \geq \left(\frac{n}{2 \log n}\right)^{\log n} = 2^{\log n \cdot \log(n/(2 \log n))}.$$

Since $\log(n/(2 \log n)) = \log n - \log(2 \log n) = \log n - O(\log \log n) = \Theta(\log n)$, we obtain

$$\binom{n}{k} \geq 2^{\Omega((\log n)^2)}.$$

By a fundamental result in extension complexity theory, specifically the connection between extension complexity and communication complexity established by Yannakakis, the extension complexity of a polytope P is lower bounded by the nonnegative rank of its slack matrix. For polytopes arising from combinatorial optimization problems, this can be related to the number of vertices and facets.

For the polytope $P_k(n)$, we apply the result that the extension complexity is at least the logarithm of the nonnegative rank of the slack matrix. Using techniques from communication complexity and the specific structure of clique polytopes, it has been shown that for appropriate choices of k, the slack matrix has nonnegative rank at least exponential in n.

Specifically, for $k = \lceil \log n \rceil$, the polytope $P_k(n)$ can be related to the correlation polytope and the cut polytope, for which exponential lower bounds on extension complexity are known. By the reduction argument and the fact that the slack matrix of $P_k(n)$ has dimensions $\binom{n}{k} \times m$ where m is the number of facets, and using the lower bound on $\binom{n}{k}$ established above, we obtain that the nonnegative rank is at least $2^{\Omega(n)}$.

Therefore, the extension complexity of $P_k(n)$ satisfies

$$\text{xc}(P_k(n)) \geq 2^{\Omega(n)}.$$

This completes the proof that there exists $k = \lceil \log n \rceil$ for which the convex hull of cliques of size k has extension complexity $2^{\Omega(n)}$. □

M.9.4 Gemini

Proof. Let n be an even integer and let $m = n/2$. We choose $k = m$. Let $Q_k(n) \subseteq \mathbb{R}^{\binom{n}{2}}$ denote the convex hull of the characteristic vectors of all cliques of size k in the complete graph K_n. We identify the vertices of K_n with the set $V = \{1, \ldots, m\} \cup \{m+1, \ldots, 2m\}$.

We rely on the lower bound for the extension complexity of the Boolean Quadric Polytope, denoted BQP_m. The polytope $BQP_m \subseteq \mathbb{R}^{\binom{m}{2}}$ is defined as the convex hull of vectors $y \in \{0,1\}^{\binom{m}{2}}$ such that there exists a vector $x \in \{0,1\}^m$ satisfying $y_{\{i,j\}} = x_i x_j$ for all $1 \le i < j \le m$. It has been established by Fiorini et al. (2015) that the extension complexity of the correlation polytope, and consequently its projection BQP_m, satisfies:

$$xc(BQP_m) \ge 2^{\Omega(m)}.$$

We show that BQP_m is a linear projection of a face of $Q_k(n)$. Consider the pairs of vertices $P_i = \{i, m+i\}$ for $i = 1, \ldots, m$. Let $z \in \mathbb{R}^{\binom{n}{2}}$ be the variables of $Q_k(n)$ indexed by edges of K_n. We define a face F of $Q_k(n)$ by imposing the constraints that the edges within each pair P_i are not selected:

$$F = Q_k(n) \cap \{z \in \mathbb{R}^{\binom{n}{2}} \mid z_{\{i,m+i\}} = 0 \text{ for all } i = 1, \ldots, m\}.$$

Let C be a clique of size $k = m$ in K_n such that its characteristic vector χ^C lies in F. The condition $z_{\{i,m+i\}} = 0$ implies that the edge $\{i, m+i\}$ is not in the clique. Since C is a clique, it cannot contain both endpoints of any edge not in the clique; thus, $|C \cap P_i| \le 1$ for all i. Furthermore, since $|C| = m$ and the pairs P_i form a partition of V into m sets, it must be that $|C \cap P_i| = 1$ for all $i = 1, \ldots, m$.

We define a linear projection $\pi : \mathbb{R}^{\binom{n}{2}} \to \mathbb{R}^{\binom{m}{2}}$ mapping a vector z to its restriction on the coordinates corresponding to edges $\{i, j\}$ where $1 \le i < j \le m$.

Consider a vertex v of BQP_m associated with a vector $x \in \{0,1\}^m$. We construct a set $S_x \subset V$ by selecting i if $x_i = 1$ and $m+i$ if $x_i = 0$. The set S_x has size m and satisfies $|S_x \cap P_i| = 1$. Thus, S_x forms a clique in K_n whose characteristic vector χ^{S_x} lies in F. For any $1 \le i < j \le m$, the edge $\{i, j\}$ is in the clique S_x if and only if $i \in S_x$ and $j \in S_x$, which is equivalent to $x_i = 1$ and $x_j = 1$. Therefore, $\pi(\chi^{S_x})_{\{i,j\}} = x_i x_j$, implying $\pi(\chi^{S_x}) = v$.

Since every vertex of BQP_m is the image of a vertex of F, and every vertex of F maps to a valid vertex of BQP_m, BQP_m is a linear projection of the face F. The extension complexity is non-increasing under taking faces and linear projections. Therefore:

$$xc(Q_k(n)) \ge xc(F) \ge xc(BQP_m) \ge 2^{\Omega(m)}.$$

Substituting $m = n/2$, we obtain $xc(Q_{n/2}(n)) \ge 2^{\Omega(n)}$. Thus, there exists a k (specifically $k = n/2$) for which the extension complexity is exponential in n. □

M.10 Problem 12.10

M.10.1 Problem Statement

Theorem M.11. *The cut polytope $\mathcal{K}_n$ is the convex hull of all cuts in a graph. Here the number of variables is $\binom{n}{2}$, one variable for each potential edge $e \subset [n]$ of size 2. For every set $A \subseteq [n]$ define the vertex*

$$y_e^A = \begin{cases} 1 & \text{if } |e \cap A| = 1, \\ 0 & \text{otherwise.} \end{cases}$$

The cut polytope is the convex hull of all these points.

1. How many vertices does $\mathcal{K}_n$ have?

2. Prove that the extension complexity of the cut polytope is $2^{\Omega(n)}$. Hint: Find an invertible linear map that maps the cut polytope to the correlation polytope.

M.10.2 GPT

Proof. Let $E = \binom{[n]}{2}$ and, for $A \subseteq [n]$, let $y^A \in \mathbb{R}^E$ be defined by $y^A_{\{i,j\}} = 1$ if and only if $|\{i,j\} \cap A| = 1$, and 0 otherwise.

First, for every $A \subseteq [n]$ one has $y^A = y^{[n]\setminus A}$, since $|\{i,j\} \cap A| = 1$ holds if and only if $|\{i,j\} \cap ([n] \setminus A)| = 1$. Hence there are at most 2^{n-1} distinct points among $\{y^A : A \subseteq [n]\}$. Conversely, if $A, B \subseteq [n]$ and $y^A = y^B$, then either $A = B$ or $A = [n] \setminus B$. Indeed, assume $A \neq B$ and pick $v \in A \triangle B$. If there exists $w \notin A \triangle B$, then $w \in A \cap B$ or $w \notin A \cup B$, and in either case $y^A_{\{v,w\}} = 1$ while $y^B_{\{v,w\}} = 0$ or vice versa, contradicting $y^A = y^B$. Therefore $A \triangle B = [n]$, i.e. $B = [n] \setminus A$. Thus the set of distinct cut vectors is in bijection with the equivalence classes $\{A, [n] \setminus A\}$, which are 2^{n-1} in number. Since $\mathcal{K}_n = \operatorname{conv}\{y^A : A \subseteq [n]\}$, it follows that $\mathcal{K}_n$ has exactly 2^{n-1} vertices.

Next set $m = n - 1$. Let $\mathcal{C}_m$ denote the correlation polytope in coordinates indexed by $\{(i,i) : i \in [m]\} \cup \{(i,j) : 1 \le i < j \le m\}$, namely

$$\mathcal{C}_m = \operatorname{conv}\{(x_i, x_i x_j)_{1 \le i \le m,\, 1 \le i < j \le m} : x \in \{0,1\}^m\} \subseteq \mathbb{R}^{m + \binom{m}{2}}.$$

Define an affine-linear map $T : \mathbb{R}^{\binom{n}{2}} \to \mathbb{R}^{m+\binom{m}{2}}$ by declaring, for $y \in \mathbb{R}^E$,

$$(T(y))_{i,i} = y_{\{i,n\}} \quad \text{for } 1 \le i \le m, \qquad (T(y))_{i,j} = \frac{y_{\{i,n\}} + y_{\{j,n\}} - y_{\{i,j\}}}{2} \quad \text{for } 1 \le i < j \le m.$$

Define also a linear map $S : \mathbb{R}^{m+\binom{m}{2}} \to \mathbb{R}^{\binom{n}{2}}$ by, for $(u_{i,i}, u_{i,j})$ in the domain,

$$(S(u))_{\{i,n\}} = u_{i,i} \quad \text{for } 1 \le i \le m, \qquad (S(u))_{\{i,j\}} = u_{i,i} + u_{j,j} - 2u_{i,j} \quad \text{for } 1 \le i < j \le m.$$

A direct substitution shows $S(T(y)) = y$ for all $y \in \mathbb{R}^E$, because for $1 \le i \le m$ one has $(S(T(y)))_{\{i,n\}} = (T(y))_{i,i} = y_{\{i,n\}}$, and for $1 \le i < j \le m$ one has

$$(S(T(y)))_{\{i,j\}} = (T(y))_{i,i} + (T(y))_{j,j} - 2(T(y))_{i,j} = y_{\{i,n\}} + y_{\{j,n\}} - \big(y_{\{i,n\}} + y_{\{j,n\}} - y_{\{i,j\}}\big) = y_{\{i,j\}}.$$

Similarly, $T(S(u)) = u$ for all u in the codomain, since for $1 \le i \le m$ one has $(T(S(u)))_{i,i} = (S(u))_{\{i,n\}} = u_{i,i}$, and for $1 \le i < j \le m$ one has

$$(T(S(u)))_{i,j} = \frac{(S(u))_{\{i,n\}} + (S(u))_{\{j,n\}} - (S(u))_{\{i,j\}}}{2} = \frac{u_{i,i} + u_{j,j} - \big(u_{i,i} + u_{j,j} - 2u_{i,j}\big)}{2} = u_{i,j}.$$

Thus T is an invertible affine-linear map with inverse S.

To identify the image of $\mathcal{K}_n$ under T, fix $A \subseteq [n]$ and let $B = A$ if $n \notin A$, and $B = [n] \setminus A$ if $n \in A$. Then $n \notin B$ and $y^A = y^B$, so it suffices to consider B with $n \notin B$. Define $x \in \{0,1\}^m$ by $x_i = \mathbf{1}_{\{i \in B\}}$ for $1 \le i \le m$. For such B, one has $y^B_{\{i,n\}} = 1$ if and only if $i \in B$, hence $(T(y^B))_{i,i} = y^B_{\{i,n\}} = x_i$. For $1 \le i < j \le m$, consider cases. If $x_i = x_j = 1$, then $i, j \in B$, so $y^B_{\{i,j\}} = 0$ and $(T(y^B))_{i,j} = (1 + 1 - 0)/2 = 1 = x_i x_j$. If $x_i \neq x_j$, then exactly one of i, j lies in B, so $y^B_{\{i,j\}} = 1$ and $(T(y^B))_{i,j} = (1 + 0 - 1)/2 = 0 = x_i x_j$. If $x_i = x_j = 0$, then $i, j \notin B$, so $y^B_{\{i,j\}} = 0$ and $(T(y^B))_{i,j} = (0 + 0 - 0)/2 = 0 = x_i x_j$. Therefore $T(y^A) = T(y^B) = (x_i, x_i x_j)$, which is a vertex of $\mathcal{C}_m$. Conversely, given any $x \in \{0,1\}^m$, define $B = \{i \in [m] : x_i = 1\}$ and note $n \notin B$; then the same computation yields $T(y^B) = (x_i, x_i x_j)$. Hence T maps the vertex set of $\mathcal{K}_n$ onto the vertex set of $\mathcal{C}_m$, and by convexity,

$$T(\mathcal{K}_n) = \operatorname{conv}\{T(y^A) : A \subseteq [n]\} = \operatorname{conv}\{(x_i, x_i x_j) : x \in \{0,1\}^m\} = \mathcal{C}_m.$$

Extension complexity is invariant under invertible affine transformations: if $P \subseteq \mathbb{R}^d$ and $Q \subseteq \mathbb{R}^d$ satisfy $Q = U(P)$ for an invertible affine map U, then an extended formulation of P can be transported to one of Q by applying U to the projection, and conversely by U^{-1}, so $\operatorname{xc}(P) = \operatorname{xc}(Q)$. Applying this with $U = T$ gives $\operatorname{xc}(\mathcal{K}_n) = \operatorname{xc}(\mathcal{C}_{n-1})$.

It is a known theorem that the correlation polytope has exponential extension complexity: there exists a constant $c > 0$ such that $\mathrm{xc}(\mathcal{C}_m) \geq 2^{cm}$ for all m, proved for example via nonnegative rank lower bounds for slack matrices in the work of Fiorini, Massar, Pokutta, Tiwary, and de Wolf (2012) and subsequent refinements. Therefore

$$\mathrm{xc}(\mathcal{K}_n) = \mathrm{xc}(\mathcal{C}_{n-1}) \geq 2^{c(n-1)} = 2^{\Omega(n)}.$$

This establishes the stated lower bound on the extension complexity of the cut polytope. □

M.10.3 Claude

Proof. We prove both parts of the theorem.

Part 1: Number of vertices of $\mathcal{K}_n$.

The vertices of the cut polytope $\mathcal{K}_n$ correspond to cuts in the complete graph K_n. Each cut is determined by a partition of $[n]$ into two sets A and $\overline{A} = [n] \setminus A$.

For each subset $A \subseteq [n]$, we obtain a vertex $y^A \in \mathbb{R}^{\binom{n}{2}}$ where the coordinate corresponding to edge e is $y_e^A = 1$ if $|e \cap A| = 1$ and $y_e^A = 0$ otherwise.

Note that the cuts corresponding to A and $\overline{A}$ are identical, since an edge $e = \{i, j\}$ crosses the cut if and only if exactly one of i, j is in A, which occurs if and only if exactly one of i, j is in $\overline{A}$. Therefore $y^A = y^{\overline{A}}$.

The total number of subsets of $[n]$ is 2^n. Since each cut is counted twice (once for A and once for $\overline{A}$), and we must account for the fact that $A = \emptyset$ and $A = [n]$ give the same cut (the empty cut with all coordinates zero), we have:

$$\text{Number of vertices} = \frac{2^n}{2} = 2^{n-1}.$$

Part 2: Extension complexity lower bound.

We establish a linear relationship between the cut polytope $\mathcal{K}_n$ and the correlation polytope COR_n, then use known lower bounds for the correlation polytope.

The correlation polytope COR_n is defined as the convex hull of vectors $z \in \{-1, 1\}^{\binom{n}{2}}$ where for each subset $S \subseteq [n]$, we have $z_e^S = \sigma_i^S \cdot \sigma_j^S$ for $e = \{i, j\}$, where $\sigma_i^S \in \{-1, 1\}$ is the sign assignment.

Define the linear map $\phi : \mathbb{R}^{\binom{n}{2}} \to \mathbb{R}^{\binom{n}{2}}$ by:

$$\phi(y)_e = 1 - 2y_e$$

for each edge e.

This map is invertible with inverse $\phi^{-1}(z)_e = \frac{1-z_e}{2}$.

For a cut determined by $A \subseteq [n]$, define $\sigma_i = 1$ if $i \in A$ and $\sigma_i = -1$ if $i \notin A$. Then for edge $e = \{i, j\}$:

$$\sigma_i \cdot \sigma_j = \begin{cases} 1 & \text{if } i, j \text{ are on the same side of the cut} \\ -1 & \text{if } i, j \text{ are on opposite sides of the cut} \end{cases}$$

This equals $1 - 2y_e^A$ since $y_e^A = 1$ when $|e \cap A| = 1$ (opposite sides) and $y_e^A = 0$ when $|e \cap A| \neq 1$ (same side).

Therefore, ϕ maps vertices of $\mathcal{K}_n$ bijectively to vertices of COR_n, and since ϕ is linear and invertible, it maps $\mathcal{K}_n$ to COR_n as polytopes.

The extension complexity satisfies $\mathrm{xc}(\mathcal{K}_n) = \mathrm{xc}(\mathrm{COR}_n)$ under affine transformations.

By the result of Fiorini, Massar, Pokutta, Tiwary, and de Wolf (2015), the extension complexity of the correlation polytope satisfies:

$$\mathrm{xc}(\mathrm{COR}_n) = 2^{\Omega(n)}.$$

Therefore, $\mathrm{xc}(\mathcal{K}_n) = 2^{\Omega(n)}$. □

M.10.4 Gemini

Proof. We address the two parts of the theorem sequentially.

1. Number of vertices of $\mathcal{K}_n$

Let V be the set of points $\{y^A \mid A \subseteq [n]\}$ used to define the cut polytope $\mathcal{K}_n = \text{conv}(V)$. Since each $y^A \in \{0,1\}^{\binom{n}{2}}$, the set V is a subset of the vertices of the unit hypercube. The vertices of any polytope contained in the vertex set of a hypercube are exactly the defining points themselves, as no vertex of a hypercube can be written as a convex combination of other vertices. Thus, every distinct vector in V is a vertex of $\mathcal{K}_n$. We must count the number of distinct vectors in V.

Consider the mapping $\psi : \mathcal{P}([n]) \to \{0,1\}^{\binom{n}{2}}$ defined by $\psi(A) = y^A$. For any edge $e = \{i,j\}$, the component y_e^A is given by:

$$y_e^A = \begin{cases} 1 & \text{if } |\{i,j\} \cap A| = 1 \\ 0 & \text{otherwise} \end{cases}$$

Observe that for any $A \subseteq [n]$, let $A^c = [n] \setminus A$ be its complement. Then:

$$|\{i,j\} \cap A^c| = 1 \iff |\{i,j\} \cap A| = 1.$$

This implies $y^A = y^{A^c}$. Thus, the map ψ is at least 2-to-1.

To show that there are no other collisions, suppose $y^A = y^B$. This equality implies that the cut defined by A is identical to the cut defined by B. The cut edges uniquely determine the partition of the graph vertices into two sets $\{S, S^c\}$ (assuming the graph is connected, which is true for the complete graph K_n associated with the variables of $\mathcal{K}_n$). Therefore, the partition $\{A, A^c\}$ must be identical to the partition $\{B, B^c\}$. This implies either $B = A$ or $B = A^c$.

The total number of subsets $A \subseteq [n]$ is 2^n. Since the mapping ψ identifies exactly pairs of complementary sets, the number of distinct vertices is:

$$|V(\mathcal{K}_n)| = \frac{2^n}{2} = 2^{n-1}.$$

2. Extension Complexity of $\mathcal{K}_n$

We prove that the extension complexity $xc(\mathcal{K}_n)$ is exponential in n by establishing an affine isomorphism between $\mathcal{K}_n$ and the correlation polytope, for which exponential lower bounds are known.

Let the correlation polytope $\mathcal{C}_n$ be defined as the convex hull of all rank-1 symmetric matrices generated by vectors in $\{-1,1\}^n$:

$$\mathcal{C}_n = \text{conv}\{zz^T \mid z \in \{-1,1\}^n\} \subseteq \mathbb{R}^{n \times n}.$$

It is a celebrated result (Fiorini et al., 2015) that the extension complexity of the correlation polytope satisfies $xc(\mathcal{C}_n) \geq 2^{\Omega(n)}$.

We consider the projection of $\mathcal{C}_n$ onto the subspace of off-diagonal entries, denoted $P_{\text{off}}(\mathcal{C}_n) \subseteq \mathbb{R}^{\binom{n}{2}}$. Since the diagonal entries of any matrix zz^T for $z \in \{-1,1\}^n$ are identically 1 ($z_i^2 = 1$), the projection is an affine isomorphism between $\mathcal{C}_n$ and $P_{\text{off}}(\mathcal{C}_n)$. Consequently, $xc(P_{\text{off}}(\mathcal{C}_n)) = xc(\mathcal{C}_n)$.

The vertices of $P_{\text{off}}(\mathcal{C}_n)$ are vectors $w \in \{-1,1\}^{\binom{n}{2}}$ indexed by pairs $\{i,j\}$ with $i < j$, where $w_{ij} = z_i z_j$ for some $z \in \{-1,1\}^n$.

We define an affine map $\Phi : \mathbb{R}^{\binom{n}{2}} \to \mathbb{R}^{\binom{n}{2}}$ acting on the coordinates x_{ij} as follows:

$$\Phi(x)_{ij} = \frac{1 - x_{ij}}{2}.$$

This map is clearly invertible, with inverse $\Phi^{-1}(y)_{ij} = 1 - 2y_{ij}$.

Let us analyze the image of the vertices of $P_{\text{off}}(\mathcal{C}_n)$ under Φ. Let $z \in \{-1,1\}^n$ and let w be the corresponding vertex in $P_{\text{off}}(\mathcal{C}_n)$ with $w_{ij} = z_i z_j$. Define a set $A \subseteq [n]$ by $A = \{i \in [n] \mid z_i = -1\}$. Then for any pair $\{i,j\}$:

- If $i \in A$ and $j \in A$, then $z_i = -1, z_j = -1 \implies w_{ij} = 1$. Then $\Phi(w)_{ij} = \frac{1-1}{2} = 0$.

- If $i \notin A$ and $j \notin A$, then $z_i = 1, z_j = 1 \implies w_{ij} = 1$. Then $\Phi(w)_{ij} = \frac{1-1}{2} = 0$.
- If one is in A and the other is not, then $z_i z_j = -1 \implies w_{ij} = -1$. Then $\Phi(w)_{ij} = \frac{1-(-1)}{2} = 1$.

Comparing this to the definition of the cut vertex y^A, we see that $\Phi(w)_{ij} = 1$ if and only if $|e \cap A| = 1$, and 0 otherwise. Thus, $\Phi(w) = y^A$.

Since Φ maps the set of vertices of $P_{\text{off}}(\mathcal{C}_n)$ surjectively onto the set of vertices of $\mathcal{K}_n$, and Φ is an affine isomorphism, it maps the convex hulls to each other isomorphically. Extension complexity is invariant under affine isomorphisms. Therefore:

$$xc(\mathcal{K}_n) = xc(P_{\text{off}}(\mathcal{C}_n)) = xc(\mathcal{C}_n).$$

Using the lower bound for the correlation polytope:

$$xc(\mathcal{K}_n) \geq 2^{\Omega(n)}.$$

□

N Problems in Chapter 13

N.1 Problem 13.1

N.1.1 Problem Statement

Lemma N.1. *The graph G has at least $n|Q|$ triangles, and no two distinct triangles in G share an edge.*

The universe U is the set of triangles in G. The graph $G_{X,Y,Z}$ is the subgraph of G defined by

$$\begin{aligned}
\{a,b\} \in E(G_{X,Y,Z}) &\iff \text{a triangle of } G \text{ containing } \{a,b\} \text{ is in } Z,\\
\{b,c\} \in E(G_{X,Y,Z}) &\iff \text{a triangle of } G \text{ containing } \{b,c\} \text{ is in } X,\\
\{a,c\} \in E(G_{X,Y,Z}) &\iff \text{a triangle of } G \text{ containing } \{a,c\} \text{ is in } Y.
\end{aligned}$$

Lemma N.2. *The graph $G_{X,Y,Z}$ contains a triangle if and only if $X \cap Y \cap Z \neq \varnothing$.*

N.1.2 GPT

Proof. Let G be a graph and let U denote the set of all triangles in G. Assume that G has at least $n|Q|$ triangles and that no two distinct triangles in G share an edge. Fix subsets $X, Y, Z \subseteq U$ and define the graph $G_{X,Y,Z}$ as follows: for any two vertices $u, v \in V(G)$ with $\{u,v\} \in E(G)$, there exists a unique triangle of G containing the edge $\{u,v\}$, by the assumption that no two distinct triangles share an edge. Denote this unique triangle by $T(\{u,v\}) \in U$. Then the definition of $G_{X,Y,Z}$ is equivalently

$$\{a,b\} \in E(G_{X,Y,Z}) \iff T(\{a,b\}) \in Z, \quad \{b,c\} \in E(G_{X,Y,Z}) \iff T(\{b,c\}) \in X, \quad \{a,c\} \in E(G_{X,Y,Z}) \iff T$$

where the notation indicates that if the unordered pair is of the corresponding form then membership is determined by the indicated set, and if the pair is not an edge of G then it is not an edge of $G_{X,Y,Z}$.

We prove that $G_{X,Y,Z}$ contains a triangle if and only if $X \cap Y \cap Z \neq \varnothing$.

First assume that $X \cap Y \cap Z \neq \varnothing$. Choose $T \in X \cap Y \cap Z$. Since T is a triangle of G, there exist distinct vertices $a, b, c \in V(G)$ such that $\{a,b\}, \{b,c\}, \{a,c\} \in E(G)$ and $T = \{a,b,c\}$. For each of these three edges, the unique triangle of G containing it is exactly T, hence $T(\{a,b\}) = T(\{b,c\}) = T(\{a,c\}) = T$. Because $T \in Z$, it follows that $\{a,b\} \in E(G_{X,Y,Z})$; because $T \in X$, it follows that $\{b,c\} \in E(G_{X,Y,Z})$; and because $T \in Y$, it follows that $\{a,c\} \in E(G_{X,Y,Z})$. Therefore the vertices a, b, c form a triangle in $G_{X,Y,Z}$.

Conversely, assume that $G_{X,Y,Z}$ contains a triangle. Then there exist distinct vertices $a, b, c \in V(G)$ such that $\{a,b\}, \{b,c\}, \{a,c\} \in E(G_{X,Y,Z})$. In particular, $\{a,b\}, \{b,c\}, \{a,c\} \in E(G)$. Hence $\{a,b,c\}$ spans a triangle in G, and thus the triangle $T_0 := \{a,b,c\}$ lies in U. By uniqueness of the triangle containing a given edge, we have $T(\{a,b\}) = T_0$, $T(\{b,c\}) = T_0$, and $T(\{a,c\}) = T_0$. Since $\{a,b\} \in E(G_{X,Y,Z})$, the definition gives $T(\{a,b\}) \in Z$, so $T_0 \in Z$. Since $\{b,c\} \in E(G_{X,Y,Z})$, the definition gives $T(\{b,c\}) \in X$, so $T_0 \in X$. Since $\{a,c\} \in E(G_{X,Y,Z})$, the definition gives $T(\{a,c\}) \in Y$, so $T_0 \in Y$. Therefore $T_0 \in X \cap Y \cap Z$, and thus $X \cap Y \cap Z \neq \varnothing$.

This proves that $G_{X,Y,Z}$ contains a triangle if and only if $X \cap Y \cap Z \neq \varnothing$. □

N.1.3 Claude

Proof. We prove both directions of the biconditional statement.

Forward direction: Suppose $G_{X,Y,Z}$ contains a triangle. Let the vertices of this triangle be a, b, and c. By the definition of $G_{X,Y,Z}$, we have:

(i) $\{a, b\} \in E(G_{X,Y,Z})$, which means there exists a triangle $T_1 \in Z$ in G containing the edge $\{a, b\}$.

(ii) $\{b, c\} \in E(G_{X,Y,Z})$, which means there exists a triangle $T_2 \in X$ in G containing the edge $\{b, c\}$.

(iii) $\{a, c\} \in E(G_{X,Y,Z})$, which means there exists a triangle $T_3 \in Y$ in G containing the edge $\{a, c\}$.

Since the triangle with vertices a, b, c exists in $G_{X,Y,Z}$, all three edges $\{a, b\}$, $\{b, c\}$, and $\{a, c\}$ must be present. By the first lemma, no two distinct triangles in G share an edge. Therefore, any triangle in G containing edge $\{a, b\}$ must also contain the third vertex that completes the triangle with a and b.

Consider the triangle $T_1 \in Z$ containing $\{a, b\}$. This triangle must have a third vertex, say v_1, forming the triangle $\{a, b, v_1\}$. Similarly, $T_2 \in X$ containing $\{b, c\}$ has the form $\{b, c, v_2\}$ for some vertex v_2, and $T_3 \in Y$ containing $\{a, c\}$ has the form $\{a, c, v_3\}$ for some vertex v_3.

Since $\{a, b\}$, $\{b, c\}$, and $\{a, c\}$ all appear as edges in $G_{X,Y,Z}$, and these three edges form a triangle in $G_{X,Y,Z}$, there must exist a triangle in G containing all three vertices a, b, and c. Let T denote this triangle in G.

By the uniqueness property from the first lemma (no two distinct triangles share an edge), we have:

- T is the unique triangle containing $\{a, b\}$, so $T = T_1 \in Z$.
- T is the unique triangle containing $\{b, c\}$, so $T = T_2 \in X$.
- T is the unique triangle containing $\{a, c\}$, so $T = T_3 \in Y$.

Therefore, $T \in X \cap Y \cap Z$, which implies $X \cap Y \cap Z \neq \varnothing$.

Reverse direction: Suppose $X \cap Y \cap Z \neq \varnothing$. Let $T \in X \cap Y \cap Z$ be a triangle in G. Since T is a triangle in G, it has three vertices, say a, b, and c, and three edges $\{a, b\}$, $\{b, c\}$, and $\{a, c\}$.

By the definition of $G_{X,Y,Z}$:

- Since $T \in Z$ and T contains the edge $\{a, b\}$, we have $\{a, b\} \in E(G_{X,Y,Z})$.
- Since $T \in X$ and T contains the edge $\{b, c\}$, we have $\{b, c\} \in E(G_{X,Y,Z})$.
- Since $T \in Y$ and T contains the edge $\{a, c\}$, we have $\{a, c\} \in E(G_{X,Y,Z})$.

Therefore, the vertices a, b, and c form a triangle in $G_{X,Y,Z}$, completing the proof. □

N.1.4 Gemini

Proof. Let U be the set of triangles in G. A crucial property of G given in the lemma statement is that no two distinct triangles in G share an edge. This implies that for any edge $e \in E(G)$ that belongs to a triangle, there exists a unique triangle $T(e) \in U$ such that $e \subseteq T(e)$.

($\Rightarrow$) Suppose the graph $G_{X,Y,Z}$ contains a triangle. Let the vertices of this triangle be denoted by a, b, c. By the definition of a triangle, the edges $\{a, b\}$, $\{b, c\}$, and $\{a, c\}$ are all present in $E(G_{X,Y,Z})$. Since $G_{X,Y,Z}$ is a subgraph of G, these edges are also present in G. Consequently, the vertices $\{a, b, c\}$ form a triangle in G. Let $T = \{a, b, c\}$. By definition, $T \in U$. We examine the conditions for the existence of these edges in $G_{X,Y,Z}$:

1. Since $\{a, b\} \in E(G_{X,Y,Z})$, the definition implies that the triangle in G containing $\{a, b\}$ must be in Z. The unique triangle in G containing $\{a, b\}$ is T. Therefore, $T \in Z$.
2. Since $\{b, c\} \in E(G_{X,Y,Z})$, the definition implies that the triangle in G containing $\{b, c\}$ must be in X. The unique triangle in G containing $\{b, c\}$ is T. Therefore, $T \in X$.

3. Since $\{a, c\} \in E(G_{X,Y,Z})$, the definition implies that the triangle in G containing $\{a, c\}$ must be in Y. The unique triangle in G containing $\{a, c\}$ is T. Therefore, $T \in Y$.

Since $T \in X$, $T \in Y$, and $T \in Z$, we have $T \in X \cap Y \cap Z$. Thus, $X \cap Y \cap Z \neq \varnothing$.

($\Leftarrow$) Suppose $X \cap Y \cap Z \neq \varnothing$. Let T be an element of $X \cap Y \cap Z$. Since $T \in U$, T is a triangle in G. Let the vertices of T be labeled a, b, c, so $T = \{a, b, c\}$. The edges of this triangle in G are $\{a, b\}$, $\{b, c\}$, and $\{a, c\}$. We verify that these edges exist in $G_{X,Y,Z}$:

1. Consider the edge $\{a, b\}$. The unique triangle in G containing $\{a, b\}$ is T. Since $T \in Z$, the condition for $\{a, b\} \in E(G_{X,Y,Z})$ is satisfied.

2. Consider the edge $\{b, c\}$. The unique triangle in G containing $\{b, c\}$ is T. Since $T \in X$, the condition for $\{b, c\} \in E(G_{X,Y,Z})$ is satisfied.

3. Consider the edge $\{a, c\}$. The unique triangle in G containing $\{a, c\}$ is T. Since $T \in Y$, the condition for $\{a, c\} \in E(G_{X,Y,Z})$ is satisfied.

Since all three edges $\{a, b\}$, $\{b, c\}$, and $\{a, c\}$ exist in $G_{X,Y,Z}$, the vertices a, b, c form a triangle in $G_{X,Y,Z}$. $\square$