DSR-Bench: Evaluating the Structural Reasoning Abilities of LLMs via Data Structures

Yu He [*1] **Yingxi Li** [*1] **Colin White** [2] **Ellen Vitercik** [1]

Abstract

Large language models (LLMs) are increasingly used in tasks involving complex mathematical and algorithmic reasoning. A core but often overlooked requirement across these tasks is the ability to perform *structural reasoning*—that is, to understand and reason about data relationships. For example, theorem proving requires maintaining a proof tree that organizes the hierarchical relationships among proof statements. However, existing benchmarks primarily focus on high-level, application-driven evaluations without isolating this fundamental capability. To address this gap, we introduce `DSR-Bench`, a novel benchmark evaluating LLMs' structural reasoning capabilities through data structures, which provide interpretable representations of data relationships. `DSR-Bench` includes 20 data structures, 35 operations, and 4,140 problem instances, organized hierarchically for fine-grained analysis of reasoning limitations. Our evaluation pipeline is fully automated and deterministic, eliminating subjective human or model-based judgments. We benchmark nine state-of-the-art LLMs, including some most advanced reasoning models. Our analysis shows that instruction-tuned models struggle with basic multi-attribute and multi-hop reasoning. Furthermore, while reasoning-oriented models perform better, they remain fragile on complex and hybrid structures, with the best model achieving an average score of only 47% on the `challenge` subset. Crucially, models often perform poorly on multi-dimensional data and natural language task descriptions, highlighting a critical gap for real-world deployment.

[*]Equal contribution [1]Stanford University, Stanford, CA, USA [2]Abacus.AI, San Francisco, CA, USA. Correspondence to: Yu He <heyu@stanford.edu>, Yingxi Li <yingxi@stanford.edu>.

The second AI for MATH Workshop at the 42nd International Conference on Machine Learning, Vancouver, Canada, Copyright 2025 by the author(s).

1. Introduction

Large language models (LLMs) have demonstrated impressive capabilities across practical tasks such as database management (Zhou et al., 2024b), coding (Chen et al., 2021; Zheng et al., 2023), scheduling (Lawless et al., 2024), and prominently, mathematical reasoning (Jiang et al., 2022; Hendrycks et al., 2021). These tasks differ in domain, but they share a core requirement: the ability to understand and manipulate data according to relationships such as order, hierarchy, and connectivity—a capability we refer to as *structural reasoning*.

Structural reasoning is an essential, yet often implicit, requirement in mathematical problem solving. For example, theorem proving requires building and maintaining a proof tree, where each node represents a logical statement and each edge captures a dependency. Structural reasoning enables us to break goals down recursively, track sub-proofs, and manage shared or failed branches — all of which hinge on the ability to comprehend and maintain a tree structure. The right structural representation can also simplify key algorithms: Kruskal's minimum-spanning-tree (MST) algorithm is most naturally formalized with a disjoint-set union data structure, while a priority queue underpins both the understanding and execution of Dijkstra's shortest-path algorithm. As researchers grow more interested in evaluating LLMs' mathematical reasoning ability, it becomes imperative to rigorously evaluate their structural reasoning capabilities to (i) expose potential failure modes and (ii) pave the way for further improvements in LLMs aimed at mathematical applications. Furthermore, LLMs are known to struggle with complex forms of reasoning, such as spatial reasoning (Li et al., 2024a; Zha et al., 2025), underscoring the timely need for a systematic evaluation of their structural reasoning abilities.

However, existing benchmarks offer limited insight into structural reasoning. Most evaluations focus on high-level, domain-specific tasks such as mathematical problem solving (Liu et al., 2024b;a; White et al., 2025), competitive programming (Jain et al., 2025), or STEM-related questions (Hendrycks et al., 2021). These tasks are typically multi-step and complex, making it difficult to isolate structural reasoning performance from other reasoning abilities.

As a result, a critical gap remains: there are no targeted, fine-grained benchmarks dedicated to structural reasoning, a core skill underlying many downstream applications.

To address this gap, we propose evaluating structural reasoning through the lens of *data structures*. Data structures offer a controlled and interpretable framework for modeling diverse data relationships: arrays for sequences, stacks and queues for temporal logic, trees for hierarchies, and graphs for complex networks. Since structural reasoning manifests as operations on these canonical structures, data structure tasks provide a principled way to probe LLM reasoning abilities at a granular level. Such a fine-grained evaluation can not only identify reliable applications of LLMs for safe deployment, but also pinpoint areas where reasoning breaks down and how it might be improved.

Building on this insight, we introduce `DSR-Bench`, a benchmark designed to isolate and evaluate LLM structural reasoning via data structure manipulation tasks. `DSR-Bench` comprises 4,140 problem instances spanning 20 data structures (grouped into six relationship categories, see Figure 1) and 35 operations across three length types (*short*, *medium*, and *long*). We also introduce `DSR-Bench-challenge`, a subset of `DSR-Bench` consisting of data structures with particularly complex designs. Each task probes whether the model can construct, maintain, and reason about data relationships within an interpretable and automatically verifiable setting.

`DSR-Bench` offers several key advantages for evaluating structural reasoning:

- **Hierarchical organization.** Tasks are organized by increasing structural complexity, enabling a fine-grained analysis of specific reasoning skills. Within each category, we design a range of tasks to isolate different sources of complexity, allowing structural reasoning to be broken down into progressively more challenging tasks. This approach precisely identifies the specific types of data relationships that pose difficulties for LLMs.
- **Deterministic evaluation.** Each data structure task has a concise and well-defined final state as its ground-truth label, allowing for deterministic and unambiguous scoring. Unlike open-ended tasks, this design supports fully automated evaluation without the need for human or model-based judgment, resulting in a fairer and more objective evaluation pipeline (Chiang et al., 2024; Feuer et al., 2024; Ye et al., 2024).
- **Synthetic, low-contamination data.** All tasks are generated efficiently from synthetic distributions, significantly reducing contamination risks from pretraining data, a major concern in benchmark design (Golchin & Surdeanu, 2023; Zhou et al., 2023). This setup also enables large-scale evaluation with minimal human involvement.

We benchmark nine state-of-the-art LLMs, including both instruction-tuned and reasoning models. Our analysis reveals the following key insights:

1. **Instruction-tuned models struggle with multi-attribute and multi-hop reasoning.** Their accuracy drops sharply on tasks involving multi-attribute data (such as hashmaps) and tasks requiring the maintenance of multi-hop properties (such as red-black trees). Prompting strategies that encourage intermediate steps can improve performance, but techniques like CoT tend to be most effective when the data structure is less standard.
2. **Reasoning models still struggle with complex structures and non-standard tasks.** Despite outperforming instruction-tuned models, reasoning models perform poorly on structurally complex tasks, with the best scoring only 47% on `DSR-Bench-challenge`. Some also fail when task instructions conflict with what are likely learned assumptions (e.g., consistently predicting tree depth as the number of *edges* instead of *nodes*, despite explicit instructions). These errors suggest a reliance on entrenched priors, limiting the model's ability to adapt to user-defined constraints.
3. **Model performance declines significantly on complex spatial data structures.** Accuracy drops as data dimensionality increases and degrades further when data is non-uniform. These patterns suggest a reliance on memorization over robust generalization or genuine reasoning.
4. **LLMs perform worse when data structure tasks are presented in natural language.** Translating formal task descriptions into natural language leads to a marked drop in performance, posing a barrier to the practical deployment of LLMs for real-world problem solving.

We release all code at https://github.com/dransyhe/DSR-Bench and all data at https://huggingface.co/collections/vitercik-lab/dsr-bench-6826381f6297ff1499134163 to support full reproducibility. We welcome community engagement and collaboration.

2. `DSR-Bench`: The Data Structure Reasoning Benchmark

In `DSR-Bench`, we propose a taxonomy of data structures grouped into six categories based on the types of data relationships they encode: Linear, Temporal, Associative, Hierarchical, Network, and Hybrid. This taxonomy captures diverse relational patterns found in real-world data and enables systematic evaluation of LLM capabilities across relationship types. Within each category, we include a spectrum of data structures that vary in complexity and opera-

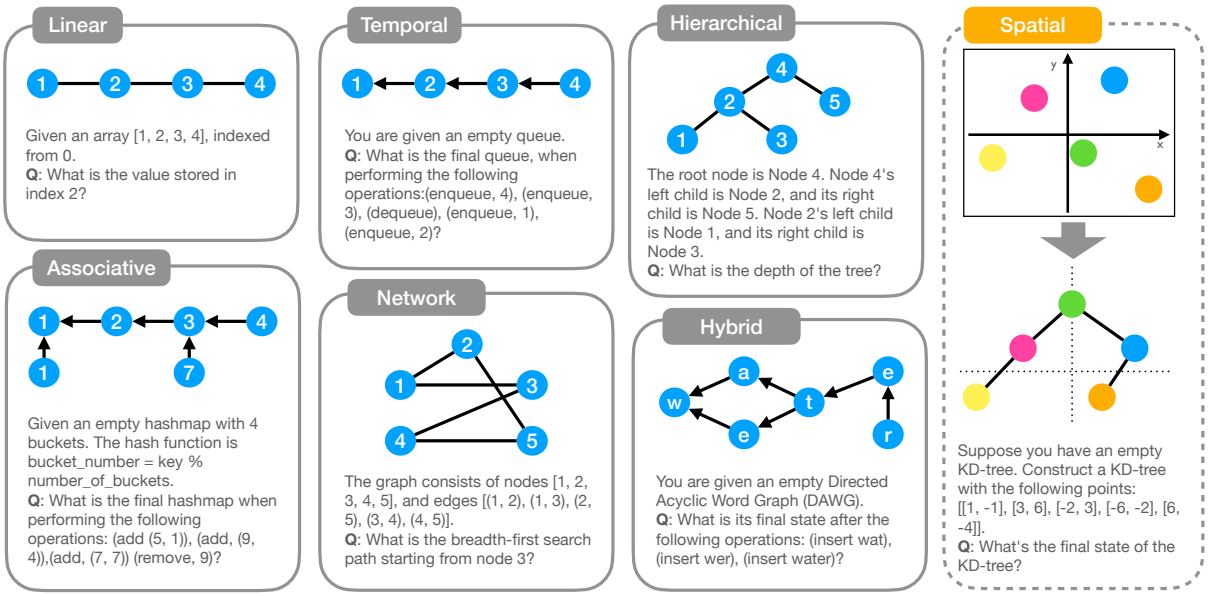

Figure 1: Overview of DSR-Bench's six data structure categories: Linear, Temporal, Associative, Hierarchical, Network, and Hybrid. Each category captures a distinct relationship and includes core tasks over canonical structures, illustrated with example operations. DSR-Bench extends some structures into multi-dimensional spaces to assess if LLMs can reason structurally over spatial data.

tional constraints, allowing us to isolate specific reasoning challenges and precisely diagnose where models struggle.

2.1. Data structure tasks

DSR-Bench includes the following data structures and tasks. A detailed description of each data structure is provided in Appendix A.2.

- **Linear (Sequential):** This category includes ARRAY and its operations: access, insert, delete, reverse, and search. Linear structures are fundamental as they introduce ordered relationships, enabling reasoning about position, sequence, and iteration, and they serve as a foundation for understanding more complex data abstractions.
- **Temporal (Time-based ordering):** Temporal structures include STACK, QUEUE, PRIORITY QUEUE, and SKIP LIST, which operate under last-in-first-out, first-in-first-out, priority-based, or probabilistic rules. Their operations are compound actions of insertions and deletions. Temporal reasoning is essential in systems that require ordered execution over time, such as event queues and schedulers, and introduces an additional ordering constraint beyond simple linear sequencing.
- **Associative (Key-value mapping):** We evaluate HASHMAP, TRIE, and SUFFIX TREE through tasks focused on construction and compound insertion-deletion operations. These tasks test the ability to handle key-based insertions, retrievals, and pattern matching in nested or hashed structures, which are core skills for efficient lookup and structured access in systems such as databases.
- **Hierarchical (Tree-like):** This group includes BINARY SEARCH TREE (BST), HEAP, RED-BLACK (RB) TREE, and B+ TREE, with tasks involving traversals and compound operations. These structures are common in file systems and database indexing. These tasks assess whether the model can maintain structural invariants, perform updates efficiently, and simulate recursive behavior.
- **Network (Connectivity and group membership):** This category includes GRAPH and DISJOINT SET UNION (DSU). Graph tasks focus on breadth- and depth-first traversal, testing a model's ability to reason about connectivity and explore many-to-many relationships, such as user interactions in social networks. DSU tasks centre on union-find operations, which are fundamental for dynamically managing group membership and identifying connected components in partitioned data.
- **Hybrid (Combined relationships):** Real-world systems often rely on hybrid data structures that combine multiple relational principles. This category includes LRU CACHE, which integrates temporal and memory constraints; BLOOM FILTER, for probabilistic set membership; and DIRECTED ACYCLIC WORD GRAPH (DAWG), a trie-like hierarchical graph. Tasks involve compound insertions and deletions, testing whether models can reason over combined constraints and generalize beyond individual structural concepts.

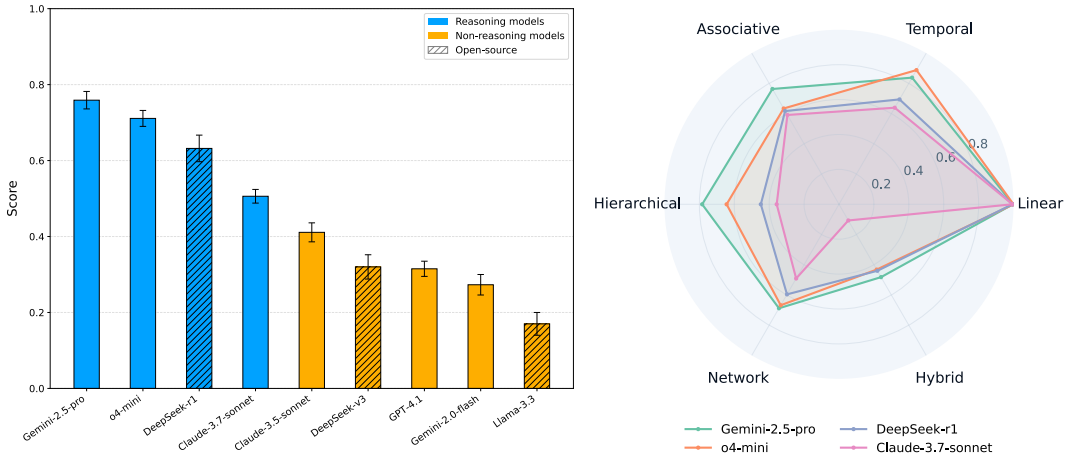

Figure 2: Left: Scores of nine models on DSR-Bench, showing standard deviations across three runs. Right: Radar chart showing how top-performing models score across six data structure categories.

2.2. Data generation

We design a diverse set of operation tasks for each data structure, as detailed in Appendix A.2. Tasks fall into three categories: construction, inspection (e.g., access, pre/in/post-order traversal), and manipulation (e.g., insert, delete). Beyond these *atomic* operations, which serve as fundamental building blocks and allow fine-grained diagnosis of specific failure modes, DSR-Bench also includes *compound* operations. These are composed of a sequence of manipulations (e.g., insert and delete) and are designed to test a model's ability to maintain structural integrity across multiple steps.

All data is synthetically generated, which is efficient and less susceptible to training data contamination (Zhang et al., 2024a; Xu et al., 2024a). Numerical inputs are sampled uniformly from the range 0 to 100, while string inputs are composed of lowercase English letters. For each task, ground-truth outputs are generated by implementing the corresponding data structure and executing the prescribed operations. The expected *format* of the final state is explicitly specified in the prompt. For simple structures like arrays, the final state is a sequence of elements. For more complex structures such as trees and graphs, the final state is represented using a canonical traversal, such as pre-order or breadth-first. The modular design of our benchmark also supports easy incorporation of new data structure types.

2.3. Suites

We categorize each task into three levels based on input length: *short* (5–10), *medium* (11–20), and *long* (21–30), to evaluate length generalization. For atomic operations, length corresponds to the number of input elements, such as list items or tree nodes. For compound operations, tasks start from an empty structure, and length is defined by the number of sequential operations applied.

We also curated a DSR-Bench-challenge subset highlighting data structures with complex designs, aimed at pushing the limits of state-of-the-art models. To further assess structural reasoning capabilities, we introduce two extension suites: DSR-Bench-spatial, focusing on spatial reasoning on multi-dimensional inputs (details see Section 4.2), and DSR-Bench-natural, evaluating reasoning over natural language descriptions (details see Section 4.3).

2.4. Prompt design

For each task, we design a prompt template and populate it with synthetic data to produce individual problem instances. Each prompt follows a consistent format: (i) a concise description of the data structure; (ii) a detailed explanation of the operations to be performed, written to avoid ambiguity; (iii) the initial state of the data structure (e.g., an existing tree or list) and any additional inputs required to execute the task (e.g., new elements to insert or delete); and (iv) a specific question requesting the final computed outcome.

Example prompt for QUEUE compound.

A queue is a data structure in which items are added at one end and removed from the other, maintaining a first-in, first-out (FIFO) order. You should create a queue. There are two types of operations: `(enqueue, k)` adds `k` to the back. `(dequeue)` removes the front.
You are given an empty queue initially.
Q: What is the final queue after performing:

- (enqueue, 49)
- (dequeue)
- ...

Answer the question in 8000 tokens.

Following recommended practices in prompt engineer-

ing (OpenAI, 2025), we also append the instruction *"Answer the question in <token> tokens"* to guide models toward producing concise outputs within a specified token budget. We also implement five different prompting methods for each task, as we detail in Section 3.2.

2.5. Structured output

A key advantage of our tasks is that model outputs can be deterministically verified using automated evaluation. This is possible because the final states of data structures can be represented as sequences of numbers or strings, enabling precise comparison. Unlike benchmarks involving proof writing or complex math problems, which yield long and ambiguous outputs requiring subjective human or LLM-based judging (Chiang et al., 2024; Feuer et al., 2024; Ye et al., 2024), our evaluation framework is more objective and reproducible.

For automatic verification, models must produce outputs matching predefined JSON schemas enforced with `Pydantic`. For instance, a Binary Search Tree (BST) pre-order traversal returns a `list[int]`. We ensure that output formats are easy to parse: about 70% of the tasks yield a 1D integer array.

2.6. Scoring system

We evaluate each model by comparing its output to the ground-truth answer. Each correct output scores 1, otherwise 0. The performance is reported as the average accuracy across all categories and length levels. To ensure deterministic outputs and eliminate ambiguity during evaluation, every task admits a single, well-defined solution. Potential sources of nondeterminism, such as definitions of hash function and tie-breaking rules, are explicitly specified in the prompt and controlled during problem generation. Additionally, we run extensive tests across all tasks and manually inspect failure cases to verify that errors are not caused by ambiguity in the question statements or task specifications.

3. Experiment settings

In this section, we describe our experimental settings. Section 3.1 outlines the models evaluated, and Section 3.2 details the prompting methods used.

3.1. Models

We evaluate a broad set of LLMs across two categories: general-purpose instruction-tuned models and models designed for advanced reasoning. From each major provider, we select the latest publicly available flagship models. Each prompt is evaluated three times, and we report both the average and standard deviation, resulting in a total of 166,230 query evaluations.

Instruction-tuned LLMs are widely used in real-world applications due to their low latency, reduced cost, and scalability. Although not explicitly trained for algorithmic or mathematical reasoning, their ability to process structured data is critical in practical settings. We test GPT-4.1-2025-04-14 (OpenAI) (OpenAI et al., 2024), Llama3.3-70B (Meta) (Grattafiori et al., 2024), Gemini-2.0-Flash-001 (Google DeepMind) (Team et al., 2024) , Claude-3-5-Sonnet-20241022 (Anthropic) (Anthropic, 2024), and DeepSeek-V3 (DeepSeek) (DeepSeek-AI et al., 2025).

We also evaluate a suite of models optimized for advanced reasoning. These models are explicitly trained to handle complex, multi-step tasks by maintaining intermediate states, including: o4-mini-2025-04-16 (OpenAI) (OpenAI, 2024), Gemini-2.5-Pro-Preview-03-25 (Google DeepMind) (Team et al., 2024), Claude-3-7-Sonnet-20250219 (Anthropic) (Anthropic, 2024), and DeepSeek-R1 (DeepSeek) (DeepSeek-AI et al., 2025).

3.2. Prompts

To examine how prompting strategies affect model performance on data structure tasks, we evaluate five configurations. Unlike reasoning models with internal mechanisms for multi-step inference (e.g., explicit reasoning tokens), instruction-tuned models are especially sensitive to prompt formulation. Our prompting configurations include: (i) **Stepwise**, which adds a "steps" field to the JSON output; (ii) **0-CoT**, which appends *"Let's think step by step"* without examples; (iii) **CoT**, which provides a single reasoning example with a final output; (iv) **3-shot**, which includes three input-output examples; and (v) **None**, the default prompting setting with no added text. See Appendix A.3 for examples.

4. Evaluation

In this section, we present experimental results evaluating all nine LLMs and highlight their strengths and weaknesses across structural reasoning tasks on `DSR-Bench` and `DSR-Bench-challenge` (Section 4.1), `DSR-Bench-spatial` (Section 4.2), and `DSR-Bench-natural` (Section 4.3).

4.1. Can LLMs understand and manipulate data structures?

In this section, we present our experimental results of LLMs on `DSR-Bench` (including `DSR-Bench-challenge`). In Section 4.1.1, we discuss the results of the five instruction-tuned models, and in Section 4.1.2, we highlight key observations on the four reasoning models.

Table 1: Scores on DSR-Bench across nine LLMs (higher is better). The table aggregates results by data structure and relationship type across 3 runs, with category scores and an overall score.

Relationship	Data Structure	o4-mini	Gemini-2.5-Pro	Claude-3.7-Sonnet	DeepSeek-R1	GPT-4.1	Gemini-2.0-Flash	Claude-3.5-Sonnet	DeepSeek-V3	Llama 3.3-70B
Linear	Array	1.00	1.00	0.99	0.99	0.94	0.90	0.96	0.98	0.69
	Category avg.	*1.00*	*1.00*	*0.99*	*0.99*	*0.94*	*0.90*	*0.96*	*0.98*	*0.69*
Temporal	Stack	1.00	1.00	0.97	0.99	0.55	0.36	0.99	0.41	0.04
	Queue	1.00	1.00	1.00	0.98	0.55	0.36	0.99	0.43	0.25
	LRU	1.00	1.00	0.30	0.16	0.85	0.78	0.82	0.01	0.50
	Priority Queue	0.55	0.51	0.28	0.65	0.25	0.16	0.33	0.20	0.08
	Category avg.	*0.89*	*0.84*	*0.64*	*0.69*	*0.55*	*0.42*	*0.79*	*0.26*	*0.22*
Associative	Hashmap	0.51	0.64	0.63	0.33	0.06	0.10	0.16	0.00	0.00
	Trie	0.68	0.80	0.08	0.49	0.18	0.17	0.49	0.02	0.00
	Suffix Tree	0.73	0.85	0.91	0.96	0.00	0.01	0.08	0.67	0.00
	Skip List	0.62	0.73	0.75	0.69	0.07	0.06	0.42	0.02	0.01
	Category avg.	*0.63*	*0.76*	*0.59*	*0.62*	*0.08*	*0.09*	*0.29*	*0.18*	*0.00*
Hierarchical	BST	0.86	0.86	0.64	0.73	0.59	0.43	0.71	0.58	0.34
	Heap	0.68	0.61	0.40	0.48	0.20	0.10	0.27	0.15	0.07
	RB tree	0.65	0.77	0.30	0.62	0.12	0.12	0.46	0.09	0.05
	B+ tree	0.97	0.95	0.38	0.31	0.23	0.12	0.23	0.08	0.01
	K-D Tree	0.47	0.71	0.00	0.45	0.00	0.01	0.03	0.00	0.00
	K-D Heap	0.10	0.05	0.05	0.11	0.04	0.04	0.05	0.03	0.01
	Category avg.	*0.64*	*0.78*	*0.33*	*0.45*	*0.20*	*0.14*	*0.34*	*0.16*	*0.08*
Network	Graph	0.87	0.94	0.11	0.67	0.15	0.05	0.06	0.06	0.02
	DSU	0.98	1.00	0.99	1.00	0.02	0.00	0.02	0.82	0.00
	Geom Graph	0.16	0.13	0.02	0.13	0.07	0.02	0.02	0.01	0.02
	Category avg.	*0.67*	*0.69*	*0.38*	*0.60*	*0.08*	*0.03*	*0.03*	*0.30*	*0.01*
Hybrid	Bloom Filter	0.61	0.89	0.16	0.74	0.04	0.04	0.04	0.04	0.02
	DAWG	0.25	0.07	0.06	0.14	0.05	0.06	0.07	0.06	0.01
	Category avg.	*0.43*	*0.48*	*0.11*	*0.44*	*0.05*	*0.05*	*0.06*	*0.05*	*0.02*
Score	***Overall avg.***	**0.72**	**0.76**	**0.51**	**0.63**	**0.31**	**0.27**	**0.41**	**0.32**	**0.17**

4.1.1. INSTRUCTION-TUNED MODELS

Instruction-tuned models struggle with multi-attribute reasoning. As shown in Table 1, these models show sharp accuracy drops in tasks involving elements with multiple attributes. For instance, while they perform well on QUEUE, their accuracy drops 30-50% on PRIORITY QUEUE, where each element includes a priority. Similarly, in the HASHMAP task, inspection of errors shows that models confuse keys and values, delete the wrong items, or hallucinate entries. These results reveal a key limitation for deployment, where managing entities with multiple interacting properties, such as deadlines, priorities, or key-value records, is common.

Multi-hop reasoning in hierarchical or network structures remains a key challenge. As Table 1 shows, while models perform reliably on BINARY SEARCH TREES, their accuracy drops by 30% on RED-BLACK TREES, revealing the difficulty of handling multi-hop properties such as keeping balance across multiple ancestral levels. Performance declines further on B+ TREES that requires reasoning over wider spans of child pointers, and on GRAPH traversal tasks with many-to-many relationships. These results underscore a persistent limitation: instruction-tuned LLMs struggle to manipulate context beyond local neighborhoods.

Prompting can help, but only when carefully designed. As shown in Table 2, the **None** prompt performs worst, suggesting that prompts encouraging stepwise reasoning can be beneficial. Our findings indicate two practical strategies: (i) Lightweight prompts such as **Stepwise** and **0-CoT** are easily implemented and improve performance (Appendix A.5); (ii) Structured prompts like **CoT** and **3-shot** are most effective for uncommon data structures. A representative case is SUFFIX-TREE CONSTRUCTION: across all models, zero-shot accuracy is below 0.40, but a well-designed CoT prompt doubles accuracy for three models (Appendix A.5).

Table 2: Average scores across all tasks for instruction-tuned models under different prompting strategies.

Model	Stepwise	0-CoT	CoT	3-shot	None
DeepSeek-V3	0.67	0.67	0.66	0.64	0.55
Llama-3.3	0.46	0.46	0.48	0.46	0.34
GPT-4.1	0.82	0.83	0.94	0.80	0.59
Gemini-2.0-Flash	0.57	0.86	0.59	0.87	0.58
Claude-3.5-Sonnet	0.72	0.92	0.72	0.93	0.69

4.1.2. REASONING MODELS

Reasoning models remain brittle on complex and spatial data structures. As shown in Table 1, reasoning models outperform instruction-tuned models in general, especially on hierarchical and networked structures. However, overall score remains below 0.5 on DSR-Bench-challenge (Table 3), in particular for *long* tasks and complex data structures. For example, the highest score on HEAP is only 0.62, despite it being a widely used structure in

Table 3: Scores on the `DSR-Bench-challenge` suite for four reasoning models.

Model	Score	Priority Queue	Suffix Tree	Trie	Skip List	Heap	Red-black tree	B+ Tree	K-D Tree	K-D Heap	DSU	Geom Graph	Bloom Filter	DAWG
Gemini-2.5-Pro	0.47	0.24	0.89	0.59	0.61	0.62	0.60	0.96	0.67	0.00	0.99	0.00	0.69	0.00
DeepSeek-R1	0.36	0.48	0.90	0.05	0.54	0.27	0.37	0.21	0.01	0.01	0.99	0.01	0.31	0.00
o4-mini	0.34	0.30	0.37	0.32	0.41	0.71	0.37	0.94	0.38	0.00	0.94	0.00	0.07	0.06
Claude-3.7-Sonnet	0.21	0.04	0.79	0.00	0.61	0.13	0.12	0.13	0.00	0.00	0.98	0.00	0.00	0.00

scheduling and resource allocation. Notably, accuracy on K-D TREE, K-D HEAP, and GEOMETRIC GRAPHS is low even for *short* tasks, suggesting that high-dimensional spatial reasoning remains a significant challenge (Appendix A.4). To further probe these limitations, we introduce `DSR-Bench-spatial`, described in Section 4.2.

Implicit priors may hinder instruction following. We observe that while GPT-4.1 scores 0.66 on the BINARY SEARCH TREE depth task (*short*), o4-mini performs much worse, with a score of just 0.01 (Appendix A.4). Manual inspection of incorrect outputs reveals that o4-mini consistently interprets depth as the number of *edges* rather than *nodes*, despite explicit instructions. Similarly, in an ablation on K-D HEAP (Table 4), switching from lexicographic order to Euclidean norm causes o4-mini's score to drop by over 0.40, as it continues to apply lexicographic comparisons. Directly querying o4-mini about the default implementation of a K-D heap confirms this point: it assumes lexicographic keys. These results suggest that reasoning models such as o4-mini may struggle to override entrenched priors, limiting their reliability on tasks with strict user-defined constraints.

4.2. Can LLMs reason structurally on spatial data?

Real-world data is often represented in multi-dimensional feature spaces. To assess whether LLMs can reason over such spatial data, we extend the benchmark with `DSR-Bench-spatial`, which includes three multi-dimensional variants: K-D HEAP, K-D TREE, and GEOMETRIC GRAPH embedded in Euclidean space. These structures are common in practice; for instance, K-D trees are key data structures in computer vision and graphics. Given task complexity, we use GPT-4.1 with the **Stepwise** prompt to encourage intermediate reasoning steps for Section 4.2 and Section 4.3.

Table 4: Performance on the two comparison metrics for K-D HEAP with different k values.

k	Lexicographic	Euclidean
1	0.79	0.82
2	0.87	0.34
3	0.84	0.26
5	0.90	0.21

Performance degrades as input dimensionality increases. As shown in Table 5, accuracy consistently drops for both models as dimensionality increases. Higher-dimensional data challenges models to reason over more complex distance metrics and partitions, limiting their effectiveness in spatial tasks. For instance, K-D trees are used to expedite nearest neighbor queries over 128-dimensional SIFT descriptors in computer vision (Silpa-Anan & Hartley, 2008). However, a notable exception is the GEOMETRIC GRAPH, where 2D data performs best, likely because it is more commonly found in textbook examples used as training data.

Table 5: Scores for the three spatial data structures with input data of varying dimensionality ($k = 1, 2, 3, 5$).

	K-D Heap		K-D Tree		Geom. Graph	
k	GPT-4.1	o4-mini	GPT-4.1	o4-mini	GPT-4.1	o4-mini
1	0.74	0.82	0.91	0.82	0.18	0.19
2	0.30	0.34	0.86	0.69	0.33	0.36
3	0.26	0.26	0.92	0.68	0.18	0.23
5	0.21	0.21	0.73	0.64	0.22	0.27

Limited robustness to non-uniform data distributions. We assess LLM robustness to distribution shifts by comparing performance on uniformly sampled versus skewed or clustered data. Specifically, we test K-D tree construction tasks using three non-uniform distributions from scikit-learn (Pedregosa et al., 2011): circles, moons, and blobs (examples shown in Figure 3, Appendix A.6). As shown in Table 6, GPT-4.1's performance drops sharply on non-uniform inputs, possibly due to a higher likelihood of uniformly distributed examples in training data. Since task difficulty is held constant, this gap suggests a reliance on pattern memorization rather than true reasoning. In contrast, o4-mini shows a smaller drop, indicating that reasoning models may generalize better to distribution shifts.

Table 6: Scores on K-D TREE with varying input data distributions.

Distribution	GPT-4.1	o4-mini
Uniform	0.86	0.69
Moon	0.42	0.62
Blob	0.33	0.62
Circle	0.31	0.67

4.3. Can LLMs reason structurally on natural language?

While the previous sections evaluated LLMs on canonical data structures, real-world use-cases are often expressed in natural language. To bridge this gap, we extend the benchmark with `DSR-Bench-natural`, which embeds data structure tasks in narrative, real-world contexts, allowing us to test whether LLMs generalize structural reasoning beyond formal descriptions.

We design three real-world scenarios that implicitly require data structures: QUEUE (children buying ice cream), BINARY SEARCH TREE (clinic appointments), and GRAPH (galaxy traveling). Synthetic data follows the same distributions as Section 2.2, with realistic substitutions (e.g., names for integers). Each scenario was written by humans, paraphrased by GPT-4o. All prompts were reviewed by three annotators for clarity and unambiguity. Details of prompt generation can be found in Appendix A.7.

Table 7: Model performance on formal and natural descriptors.

Model	Task	Formal	Natural
GPT-4.1	Queue	1.00	0.77
	BST	0.88	0.59
	Graph	0.42	0.43
o4-mini	Queue	1.00	0.83
	BST	1.00	0.93
	Graph	0.84	0.67

Natural language prompt ex. for QUEUE compound.

On a sunny afternoon in the park, an ice cream truck rolled in... Children began to form a line, each newcomer taking their place at the end while the vendor served from the front...

- Isabella Miller ran over and joined the ice cream line.
- The next kid in line was served promptly.
- ...

Q: What is the order of the remaining kids in line?

LLMs struggle when shifting from formal to natural language. As shown in Table 7, performance drops when tasks are described in natural language rather than formal descriptions, despite identical operation distributions. This suggests that even reasoning models struggle to apply abilities in language-rich contexts. Bridging this gap is crucial for deployment and presents a key direction for future research. Furthermore, the higher accuracy on formal descriptors may stem from LLMs' exposure to textbook-style patterns, where integers and explicit syntax are common.

5. Related Works

Mathematical reasoning benchmarks for LLMs Many benchmarks have been proposed to evaluate the capabilities and limitations of LLMs (see Appendix A.1 for a detailed review). Mathematical reasoning-focused benchmarks have primarily focused on elementary to college-level textbook math problems (Cobbe et al., 2021; Liu et al., 2024a;b; Hendrycks et al., 2021; Glazer et al., 2024) and the capacity for logical reasoning via proof-writing (Azerbayev et al., 2022) and logic puzzles (White et al., 2025; Luo et al., 2024). These often require long, complex responses involving intertwined reasoning steps and rely on human or LLM-based judgment for evaluation (Chiang et al., 2024; Feuer et al., 2024; Ye et al., 2024). In contrast, we take a more foundational approach: testing whether LLMs can understand and manipulate data relationships, which is a core capability that underlies reasoning in math, isolating structural reasoning from other complexities.

Mathematical and algorithmic reasoning with LLMs Prior work on algorithmic reasoning has primarily focused on arithmetic tasks and length generalization in transformers (Zhou et al., 2022; 2024a; Lee et al., 2024), or on specific graph problems such as cycle detection and connectivity (Wang et al., 2023; Fatemi et al., 2024). These efforts offer limited coverage of broader data structure reasoning. One closely related work, CLRS-Text (Markeeva et al., 2024), evaluates whether LLMs can simulate 30 algorithms from the CLRS benchmark suite (Veličković et al., 2022; Cormen et al., 2009) via textual inputs. Rather than testing if LLMs can replicate specific algorithms, we evaluate their ability to reason over data structures, which are core abstractions and prerequisites for algorithmic execution. This offers a more general and interpretable assessment of algorithmic reasoning, revealing not only whether models succeed, but also which types of structural reasoning they lack.

6. Conclusion

We introduce `DSR-Bench`, the first benchmark for evaluating LLMs' structural reasoning through canonical data structures. `DSR-Bench` enables fine-grained, controlled, and objective evaluation using synthetic data and an automated scoring pipeline. Benchmarking nine state-of-the-art LLMs reveals key limitations across both instruction-tuned and reasoning models. Notably, `DSR-Bench-challenge` remains difficult even for the strongest models. In addition, we propose two extensions, `DSR-Bench-spatial` and `DSR-Bench-natural`, to evaluate spatial reasoning and natural language understanding of data structures. These results uncover further gaps, suggesting current LLMs are not yet ready for structure-dependent, real-world tasks.

Limitation and future work While our scoring system compares the final state of the data structure, additional insights could be gained by evaluating models' intermediate thought traces. However, such traces are unavailable from some providers and can be costly to retrieve. As interfaces grow more transparent, we hope to incorporate such signals to enrich error analysis and better understand model reasoning. Additionally, while `DSR-Bench` covers a wide range of structures, expanding it to cover emerging representations, such as graph-probabilistic hybrid structures, would further broaden its scope. Our benchmark is modular and extensible, and we welcome community contributions to advance these directions and help develop an even more comprehensive evaluation framework.

References

Anthropic. The claude 3 model family: Opus, sonnet, haiku. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf, 2024. Accessed: 2025-05-05.

Azerbayev, Z., Piotrowski, B., and Avigad, J. Proofnet: A benchmark for autoformalizing and formally proving undergraduate-level mathematics problems. In *Second MATH-AI Workshop*, 2022.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhang, H., Zhu, B., Jordan, M., Gonzalez, J. E., and Stoica, I. Chatbot arena: An open platform for evaluating llms by human preference, 2024.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN 0262033844.

DeepSeek-AI, Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Guo, D., Yang, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Zhang, H., Ding, H., Xin, H., Gao, H., Li, H., Qu, H., Cai, J. L., Liang, J., Guo, J., Ni, J., Li, J., Wang, J., Chen, J., Chen, J., Yuan, J., Qiu, J., Li, J., Song, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Xu, L., Xia, L., Zhao, L., Wang, L., Zhang, L., Li, M., Wang, M., Zhang, M., Zhang, M., Tang, M., Li, M., Tian, N., Huang, P., Wang, P., Zhang, P., Wang, Q., Zhu, Q., Chen, Q., Du, Q., Chen, R. J., Jin, R. L., Ge, R., Zhang, R., Pan, R., Wang, R., Xu, R., Zhang, R., Chen, R., Li, S. S., Lu, S., Zhou, S., Chen, S., Wu, S., Ye, S., Ye, S., Ma, S., Wang, S., Zhou, S., Yu, S., Zhou, S., Pan, S., Wang, T., Yun, T., Pei, T., Sun, T., Xiao, W. L., Zeng, W., Zhao, W., An, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Li, X. Q., Jin, X., Wang, X., Bi, X., Liu, X., Wang, X., Shen, X., Chen, X., Zhang, X., Chen, X., Nie, X., Sun, X., Wang, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yu, X., Song, X., Shan, X., Zhou, X., Yang, X., Li, X., Su, X., Lin, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhu, Y. X., Zhang, Y., Xu, Y., Xu, Y., Huang, Y., Li, Y., Zhao, Y., Sun, Y., Li, Y., Wang, Y., Yu, Y., Zheng, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Tang, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Wu, Y., Ou, Y., Zhu, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Zha, Y., Xiong, Y., Ma, Y., Yan, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Wu, Z. F., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Huang, Z., Zhang, Z., Xie, Z., Zhang, Z., Hao, Z., Gou, Z., Ma, Z., Yan, Z., Shao, Z., Xu, Z., Wu, Z., Zhang, Z., Li, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Gao, Z., and Pan, Z. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Fatemi, B., Halcrow, J., and Perozzi, B. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IuXR1CCrSi.

Feng, H., Ronzano, F., LaFleur, J., Garber, M., de Oliveira, R., Rough, K., Roth, K., Nanavati, J., El Abidine, K. Z., and Mack, C. Evaluation of large language model performance on the biomedical language understanding and reasoning benchmark: Comparative study. *medRxiv*, pp. 2024–05, 2024.

Feuer, B., Goldblum, M., Datta, T., Nambiar, S., Besaleli, R., Dooley, S., Cembalest, M., and Dickerson, J. P. Style outweighs substance: Failure modes of llm judges in alignment benchmarking. *arXiv preprint arXiv:2409.15268*, 2024.

Glazer, E., Erdil, E., Besiroglu, T., Chicharro, D., Chen, E., Gunning, A., Olsson, C. F., Denain, J.-S., Ho, A., Santos, E. d. O., et al. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv preprint arXiv:2411.04872*, 2024.

Golchin, S. and Surdeanu, M. Time travel in llms: Tracing data contamination in large language models. *arXiv preprint arXiv:2308.08493*, 2023.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A.,

Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X. E., Xia, X., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Le, E.-T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Caggioni, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I., Leontiadis, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Liu, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Mehta, N., Laptev, N. P., Dong, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Parthasarathy, R., Li, R., Hogan, R., Battey, R., Wang, R., Howes, R., Rinott, R., Mehta, S., Siby, S., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Mahajan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Patil, S., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Deng, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Koehler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wu, X., Wang, X., Wu, X., Gao, X., Kleinman, Y., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Zhao, Y., Hao, Y., Qian, Y., Li, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., Zhao, Z., and Ma, Z. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset.

In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Jiang, A. Q., Welleck, S., Zhou, J. P., Li, W., Liu, J., Jamnik, M., Lacroix, T., Wu, Y., and Lample, G. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. *arXiv preprint arXiv:2210.12283*, 2022.

Jiang, J., Zhou, K., Dong, Z., Ye, K., Zhao, W. X., and Wen, J.-R. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*, 2023.

Kanithi, P. K., Christophe, C., Pimentel, M. A., Raha, T., Saadi, N., Javed, H., Maslenkova, S., Hayat, N., Rajan, R., and Khan, S. Medic: Towards a comprehensive framework for evaluating llms in clinical applications. *arXiv preprint arXiv:2409.07314*, 2024.

Koo, R., Lee, M., Raheja, V., Park, J. I., Kim, Z. M., and Kang, D. Benchmarking cognitive biases in large language models as evaluators. *arXiv preprint arXiv:2309.17012*, 2023.

Lawless, C., Schoeffer, J., Le, L., Rowan, K., Sen, S., St. Hill, C., Suh, J., and Sarrafzadeh, B. "i want it that way": Enabling interactive decision support using large language models and constraint programming. *ACM Transactions on Interactive Intelligent Systems*, 14(3): 1–33, 2024.

Lee, N., Sreenivasan, K., Lee, J. D., Lee, K., and Papailiopoulos, D. Teaching arithmetic to small transformers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dsUB4bst9S.

Li, F., Hogg, D. C., and Cohn, A. G. Advancing spatial reasoning in large language models: An in-depth evaluation and enhancement using the stepgame benchmark. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18500–18507, 2024a.

Li, S., Balachandran, V., Feng, S., Ilgen, J., Pierson, E., Koh, P. W. W., and Tsvetkov, Y. Mediq: Question-asking llms and a benchmark for reliable interactive clinical reasoning. *Advances in Neural Information Processing Systems*, 37:28858–28888, 2024b.

Li, X., Chen, Z., Zhang, J. M., Lou, Y., Li, T., Sun, W., Liu, Y., and Liu, X. Benchmarking bias in large language models during role-playing. *arXiv preprint arXiv:2411.00585*, 2024c.

Liu, H., Zheng, Z., Qiao, Y., Duan, H., Fei, Z., Zhou, F., Zhang, W., Zhang, S., Lin, D., and Chen, K. Mathbench: Evaluating the theory and application proficiency of llms with a hierarchical mathematics benchmark, 2024a. URL https://arxiv.org/abs/2405.12209.

Liu, J., Zhou, P., Hua, Y., Chong, D., Tian, Z., Liu, A., Wang, H., You, C., Guo, Z., Zhu, L., et al. Benchmarking large language models on cmexam-a comprehensive chinese medical exam dataset. *Advances in Neural Information Processing Systems*, 36:52430–52452, 2023a.

Liu, S.-C., Wang, S., Chang, T., Lin, W., Hsiung, C.-W., Hsieh, Y.-C., Cheng, Y.-P., Luo, S.-H., and Zhang, J. JarviX: A LLM no code platform for tabular data analysis and optimization. In Wang, M. and Zitouni, I. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 622–630, Singapore, December 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-industry.59. URL https://aclanthology.org/2023.emnlp-industry.59/.

Liu, Y., Yao, Y., Ton, J.-F., Zhang, X., Guo, R., Cheng, H., Klochkov, Y., Taufiq, M. F., and Li, H. Trustworthy llms: a survey and guideline for evaluating large language models' alignment. *arXiv preprint arXiv:2308.05374*, 2023c.

Liu, Y., Jin, R., Shi, L., Yao, Z., and Xiong, D. Finemath: A fine-grained mathematical evaluation benchmark for chinese large language models, 2024b. URL https://arxiv.org/abs/2403.07747.

Longwell, J. B., Hirsch, I., Binder, F., Conchas, G. A. G., Mau, D., Jang, R., Krishnan, R. G., and Grant, R. C. Performance of large language models on medical oncology examination questions. *JAMA Network Open*, 7(6): e2417641–e2417641, 2024.

Luo, H., Huang, H., Deng, Z., Liu, X., Chen, R., and Liu, Z. Bigbench: A unified benchmark for social bias in text-to-image generative models based on multi-modal llm. *arXiv preprint arXiv:2407.15240*, 2024.

Markeeva, L., McLeish, S., Ibarz, B., Bounsi, W., Kozlova, O., Vitvitskyi, A., Blundell, C., Goldstein, T., Schwarzschild, A., and Veličković, P. The clrs-text algorithmic reasoning language benchmark, 2024. URL https://arxiv.org/abs/2406.04229.

Maynez, J., Agrawal, P., and Gehrmann, S. Benchmarking large language model capabilities for conditional generation. *arXiv preprint arXiv:2306.16793*, 2023.

OpenAI. O3 and o4 mini system card. https://openai.com/index/o3-o4-mini-system-card/, 2024. Accessed: 2025-05-05.

OpenAI. Six strategies for getting better results with prompt engineering. https://platform.openai.com/docs/guides/prompt-engineering/six-strategies-for-getting-better-results, 2025. Accessed: 2025-05-05.

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, J. H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorny, Pokrass, M., Pong, V. H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M. B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Silpa-Anan, C. and Hartley, R. Optimised kd-trees for fast image descriptor matching. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008. doi: 10.1109/CVPR.2008.4587638.

Singh, H., Gupta, N., Bharadwaj, S., Tewari, D., and Talukdar, P. Indicgenbench: a multilingual benchmark to evaluate generation capabilities of llms on indic languages. *arXiv preprint arXiv:2404.16816*, 2024.

Siska, C., Marazopoulou, K., Ailem, M., and Bono, J. Examining the robustness of llm evaluation to the distributional assumptions of benchmarks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10406–10421, 2024.

Sui, Y., Zhou, M., Zhou, M., Han, S., and Zhang, D. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 645–654, 2024.

Tang, X., Zong, Y., Phang, J., Zhao, Y., Zhou, W., Cohan, A., and Gerstein, M. Struc-bench: Are large language models really good at generating complex structured data? *arXiv preprint arXiv:2309.08963*, 2023.

Team, G., Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., Silver, D., Johnson, M., Antonoglou, I., Schrittwieser, J., Glaese, A., Chen, J., Pitler, E., Lillicrap, T., Lazaridou, A., Firat, O., Molloy, J., Isard, M., Barham, P. R., Hennigan, T., Lee, B., Viola, F., Reynolds, M., Xu, Y., Doherty, R., Collins, E., Meyer, C., Rutherford, E., Moreira, E., Ayoub, K., Goel, M., Krawczyk, J., Du, C., Chi, E., Cheng, H.-T., Ni, E., Shah, P., Kane, P., Chan, B., Faruqui, M., Severyn, A., Lin, H., Li, Y., Cheng, Y., Ittycheriah, A., Mahdieh, M., Chen, M., Sun, P., Tran, D., Bagri, S., Lakshminarayanan, B., Liu, J., Orban, A., Güra, F., Zhou, H., Song, X., Boffy, A., Ganapathy, H., Zheng, S., Choe, H., Ágoston Weisz, Zhu, T., Lu, Y., Gopal, S., Kahn, J., Kula, M., Pitman, J., Shah, R., Taropa, E., Merey, M. A., Baeuml, M., Chen, Z., Shafey, L. E., Zhang, Y., Sercinoglu, O., Tucker, G., Piqueras, E., Krikun, M., Barr, I., Savinov, N., Danihelka, I., Roelofs, B., White, A., Andreassen, A., von Glehn, T., Yagati, L., Kazemi, M., Gonzalez, L., Khalman, M., Sygnowski, J., Frechette, A., Smith, C., Culp, L., Proleev, L., Luan, Y., Chen, X., Lottes, J., Schucher, N., Lebron, F., Rrustemi, A., Clay, N., Crone, P., Kocisky, T., Zhao, J., Perz, B., Yu, D., Howard, H., Bloniarz, A., Rae, J. W., Lu, H., Sifre, L., Maggioni, M., Alcober, F., Garrette, D., Barnes, M., Thakoor, S., Austin, J., Barth-Maron, G., Wong, W., Joshi, R., Chaabouni, R., Fatiha, D., Ahuja, A., Tomar, G. S., Senter, E., Chadwick, M., Kornakov, I., Attaluri, N., Iturrate, I., Liu, R., Li, Y., Cogan, S., Chen, J., Jia, C., Gu, C., Zhang, Q., Grimstad, J., Hartman, A. J., Garcia, X., Pillai, T. S., Devlin, J., Laskin, M., de Las Casas, D., Valter, D., Tao, C., Blanco, L., Badia, A. P., Reitter, D., Chen, M., Brennan, J., Rivera, C., Brin, S., Iqbal, S., Surita, G., Labanowski, J., Rao, A., Winkler, S., Parisotto, E., Gu, Y., Olszewska, K., Addanki, R., Miech, A., Louis, A., Teplyashin, D., Brown, G., Catt, E., Balaguer, J., Xiang, J., Wang, P., Ashwood, Z., Briukhov, A., Webson, A., Ganapathy, S., Sanghavi, S., Kannan, A., Chang, M.-W., Stjerngren, A., Djolonga, J., Sun, Y., Bapna, A., Aitchison, M., Pejman, P., Michalewski, H., Yu, T., Wang, C., Love, J., Ahn, J., Bloxwich, D., Han, K., Humphreys, P., Sellam, T., Bradbury, J., Godbole, V., Samangooei, S., Damoc, B., Kaskasoli, A., Arnold, S. M. R., Vasudevan, V., Agrawal, S., Riesa, J., Lepikhin, D., Tanburn, R., Srinivasan, S., Lim, H., Hodkinson, S., Shyam, P., Ferret, J., Hand, S., Garg, A., Paine, T. L., Li, J., Li, Y., Giang, M., Neitz, A., Abbas, Z., York, S., Reid, M., Cole, E., Chowdhery, A., Das, D., Rogozińska, D., Nikolaev, V., Sprechmann, P., Nado, Z., Zilka, L., Prost, F., He, L., Monteiro, M., Mishra, G., Welty, C., Newlan, J., Jia, D., Allamanis, M., Hu, C. H., de Liedekerke, R., Gilmer, J., Saroufim, C., Rijhwani, S., Hou, S., Shrivastava, D., Baddepudi, A., Goldin, A., Ozturel, A., Cassirer, A., Xu, Y., Sohn, D., Sachan, D., Amplayo, R. K., Swanson, C., Petrova, D., Narayan, S., Guez, A., Brahma, S., Landon, J., Patel, M., Zhao, R., Villela, K., Wang, L., Jia, W., Rahtz, M., Giménez, M., Yeung, L., Keeling, J., Georgiev, P., Mincu, D., Wu, B., Haykal, S., Saputro, R., Vodrahalli, K., Qin, J., Cankara, Z., Sharma, A., Fernando, N., Hawkins, W., Neyshabur, B., Kim, S., Hutter, A., Agrawal, P., Castro-Ros, A., van den Driessche, G., Wang, T., Yang, F., yiin Chang, S., Komarek, P., McIlroy, R., Lučić, M., Zhang, G., Farhan, W., Sharman, M., Natsev, P., Michel, P., Bansal, Y., Qiao, S., Cao, K., Shakeri, S., Butterfield, C., Chung, J., Rubenstein, P. K., Agrawal, S., Mensch, A., Soparkar, K., Lenc, K., Chung, T., Pope, A., Maggiore, L., Kay, J., Jhakra, P., Wang, S., Maynez, J., Phuong, M., Tobin, T., Tacchetti, A., Trebacz, M., Robinson, K., Katariya, Y., Riedel, S., Bailey, P., Xiao, K., Ghelani, N., Aroyo, L., Slone, A., Houlsby, N., Xiong, X., Yang, Z., Gribovskaya, E., Adler, J., Wirth, M., Lee, L., Li, M., Kagohara, T., Pavagadhi, J., Bridgers, S., Bortsova, A., Ghemawat, S., Ahmed, Z., Liu, T., Powell, R., Bolina, V., Iinuma, M., Zablotskaia, P., Besley, J., Chung, D.-W., Dozat, T., Comanescu, R., Si, X., Greer, J., Su, G., Polacek, M., Kaufman, R. L., Tokumine, S., Hu, H., Buchatskaya, E., Miao, Y., Elhawaty, M., Siddhant, A., Tomasev, N., Xing, J., Greer, C., Miller, H., Ashraf, S., Roy, A., Zhang, Z., Ma, A., Filos, A., Besta, M., Blevins, R., Klimenko, T., Yeh, C.-K., Changpinyo, S., Mu, J., Chang, O., Pajarskas, M., Muir, C., Cohen, V., Lan, C. L., Haridasan, K., Marathe, A., Hansen, S., Douglas, S., Samuel, R., Wang, M., Austin, S., Lan, C., Jiang, J., Chiu, J., Lorenzo, J. A., Sjösund, L. L., Cevey, S., Gleicher, Z., Avrahami, T., Boral, A., Srinivasan, H., Selo, V., May, R., Aisopos, K., Hussenot, L., Soares, L. B., Baumli, K., Chang, M. B., Recasens, A., Caine, B., Pritzel, A., Pavetic, F., Pardo, F., Gergely, A., Frye, J., Ramasesh, V., Horgan, D., Badola, K., Kassner, N., Roy, S., Dyer, E., Campos, V. C., Tomala, A., Tang, Y., Badawy, D. E., White, E., Mustafa, B., Lang, O., Jindal, A., Vikram, S., Gong, Z., Caelles, S., Hemsley, R., Thornton, G., Feng, F., Stokowiec, W., Zheng, C., Thacker, P., Çağlar Ünlü, Zhang, Z., Saleh, M., Svensson, J., Bileschi, M., Patil, P., Anand, A., Ring, R., Tsihlas, K., Vezer, A., Selvi, M., Shevlane, T., Rodriguez, M., Kwiatkowski, T., Daruki, S., Rong, K., Dafoe, A., FitzGerald, N., Gu-Lemberg, K., Khan, M., Hendricks, L. A., Pellat, M., Feinberg, V., Cobon-Kerr, J., Sainath, T., Rauh, M., Hashemi, S. H., Ives, R., Hasson, Y., Noland, E., Cao, Y., Byrd, N., Hou, L., Wang, Q., Sottiaux, T., Paganini, M., Lespiau, J.-B., Moufarek, A., Hassan, S., Shivakumar, K., van Amersfoort, J., Mandhane, A., Joshi, P., Goyal, A., Tung, M., Brock, A., Sheahan, H., Misra, V., Li, C., Rakićević, N., Dehghani, M., Liu, F., Mittal, S., Oh, J., Noury, S., Sezener, E., Huot, F., Lamm, M., Cao, N. D., Chen, C., Mudgal, S., Stella, R., Brooks, K., Vasudevan, G., Liu, C., Chain, M., Melinkeri, N., Cohen, A., Wang, V., Seymore,

K., Zubkov, S., Goel, R., Yue, S., Krishnakumaran, S., Albert, B., Hurley, N., Sano, M., Mohananey, A., Joughin, J., Filonov, E., Kepa, T., Eldawy, Y., Lim, J., Rishi, R., Badiezadegan, S., Bos, T., Chang, J., Jain, S., Padmanabhan, S. G. S., Puttagunta, S., Krishna, K., Baker, L., Kalb, N., Bedapudi, V., Kurzrok, A., Lei, S., Yu, A., Litvin, O., Zhou, X., Wu, Z., Sobell, S., Siciliano, A., Papir, A., Neale, R., Bragagnolo, J., Toor, T., Chen, T., Anklin, V., Wang, F., Feng, R., Gholami, M., Ling, K., Liu, L., Walter, J., Moghaddam, H., Kishore, A., Adamek, J., Mercado, T., Mallinson, J., Wandekar, S., Cagle, S., Ofek, E., Garrido, G., Lombriser, C., Mukha, M., Sun, B., Mohammad, H. R., Matak, J., Qian, Y., Peswani, V., Janus, P., Yuan, Q., Schelin, L., David, O., Garg, A., He, Y., Duzhyi, O., Älgmyr, A., Lottaz, T., Li, Q., Yadav, V., Xu, L., Chinien, A., Shivanna, R., Chuklin, A., Li, J., Spadine, C., Wolfe, T., Mohamed, K., Das, S., Dai, Z., He, K., von Dincklage, D., Upadhyay, S., Maurya, A., Chi, L., Krause, S., Salama, K., Rabinovitch, P. G., M, P. K. R., Selvan, A., Dektiarev, M., Ghiasi, G., Guven, E., Gupta, H., Liu, B., Sharma, D., Shtacher, I. H., Paul, S., Akerlund, O., Aubet, F.-X., Huang, T., Zhu, C., Zhu, E., Teixeira, E., Fritze, M., Bertolini, F., Marinescu, L.-E., Bölle, M., Paulus, D., Gupta, K., Latkar, T., Chang, M., Sanders, J., Wilson, R., Wu, X., Tan, Y.-X., Thiet, L. N., Doshi, T., Lall, S., Mishra, S., Chen, W., Luong, T., Benjamin, S., Lee, J., Andrejczuk, E., Rabiej, D., Ranjan, V., Styrc, K., Yin, P., Simon, J., Harriott, M. R., Bansal, M., Robsky, A., Bacon, G., Greene, D., Mirylenka, D., Zhou, C., Sarvana, O., Goyal, A., Andermatt, S., Siegler, P., Horn, B., Israel, A., Pongetti, F., Chen, C.-W. L., Selvatici, M., Silva, P., Wang, K., Tolins, J., Guu, K., Yogev, R., Cai, X., Agostini, A., Shah, M., Nguyen, H., Donnaile, N. O., Pereira, S., Friso, L., Stambler, A., Kurzrok, A., Kuang, C., Romanikhin, Y., Geller, M., Yan, Z., Jang, K., Lee, C.-C., Fica, W., Malmi, E., Tan, Q., Banica, D., Balle, D., Pham, R., Huang, Y., Avram, D., Shi, H., Singh, J., Hidey, C., Ahuja, N., Saxena, P., Dooley, D., Potharaju, S. P., O'Neill, E., Gokulchandran, A., Foley, R., Zhao, K., Dusenberry, M., Liu, Y., Mehta, P., Kotikalapudi, R., Safranek-Shrader, C., Goodman, A., Kessinger, J., Globen, E., Kolhar, P., Gorgolewski, C., Ibrahim, A., Song, Y., Eichenbaum, A., Brovelli, T., Potluri, S., Lahoti, P., Baetu, C., Ghorbani, A., Chen, C., Crawford, A., Pal, S., Sridhar, M., Gurita, P., Mujika, A., Petrovski, I., Cedoz, P.-L., Li, C., Chen, S., Santo, N. D., Goyal, S., Punjabi, J., Kappaganthu, K., Kwak, C., LV, P., Velury, S., Choudhury, H., Hall, J., Shah, P., Figueira, R., Thomas, M., Lu, M., Zhou, T., Kumar, C., Jurdi, T., Chikkerur, S., Ma, Y., Yu, A., Kwak, S., Ähdel, V., Rajayogam, S., Choma, T., Liu, F., Barua, A., Ji, C., Park, J. H., Hellendoorn, V., Bailey, A., Bilal, T., Zhou, H., Khatir, M., Sutton, C., Rzadkowski, W., Macintosh, F., Shagin, K., Medina, P., Liang, C., Zhou, J., Shah, P., Bi, Y., Dankovics, A., Banga, S., Lehmann, S., Bredesen, M., Lin, Z., Hoffmann, J. E., Lai, J., Chung, R., Yang, K., Balani, N., Bražinskas, A., Sozanschi, A., Hayes, M., Alcalde, H. F., Makarov, P., Chen, W., Stella, A., Snijders, L., Mandl, M., Kärrman, A., Nowak, P., Wu, X., Dyck, A., Vaidyanathan, K., R, R., Mallet, J., Rudominer, M., Johnston, E., Mittal, S., Udathu, A., Christensen, J., Verma, V., Irving, Z., Santucci, A., Elsayed, G., Davoodi, E., Georgiev, M., Tenney, I., Hua, N., Cideron, G., Leurent, E., Alnahlawi, M., Georgescu, I., Wei, N., Zheng, I., Scandinaro, D., Jiang, H., Snoek, J., Sundararajan, M., Wang, X., Ontiveros, Z., Karo, I., Cole, J., Rajashekhar, V., Tumeh, L., Ben-David, E., Jain, R., Uesato, J., Datta, R., Bunyan, O., Wu, S., Zhang, J., Stanczyk, P., Zhang, Y., Steiner, D., Naskar, S., Azzam, M., Johnson, M., Paszke, A., Chiu, C.-C., Elias, J. S., Mohiuddin, A., Muhammad, F., Miao, J., Lee, A., Vieillard, N., Park, J., Zhang, J., Stanway, J., Garmon, D., Karmarkar, A., Dong, Z., Lee, J., Kumar, A., Zhou, L., Evens, J., Isaac, W., Irving, G., Loper, E., Fink, M., Arkatkar, I., Chen, N., Shafran, I., Petrychenko, I., Chen, Z., Jia, J., Levskaya, A., Zhu, Z., Grabowski, P., Mao, Y., Magni, A., Yao, K., Snaider, J., Casagrande, N., Palmer, E., Suganthan, P., Castaño, A., Giannoumis, I., Kim, W., Rybiński, M., Sreevatsa, A., Prendki, J., Soergel, D., Goedeckemeyer, A., Gierke, W., Jafari, M., Gaba, M., Wiesner, J., Wright, D. G., Wei, Y., Vashisht, H., Kulizhskaya, Y., Hoover, J., Le, M., Li, L., Iwuanyanwu, C., Liu, L., Ramirez, K., Khorlin, A., Cui, A., LIN, T., Wu, M., Aguilar, R., Pallo, K., Chakladar, A., Perng, G., Abellan, E. A., Zhang, M., Dasgupta, I., Kushman, N., Penchev, I., Repina, A., Wu, X., van der Weide, T., Ponnapalli, P., Kaplan, C., Simsa, J., Li, S., Dousse, O., Yang, F., Piper, J., Ie, N., Pasumarthi, R., Lintz, N., Vijayakumar, A., Andor, D., Valenzuela, P., Lui, M., Paduraru, C., Peng, D., Lee, K., Zhang, S., Greene, S., Nguyen, D. D., Kurylowicz, P., Hardin, C., Dixon, L., Janzer, L., Choo, K., Feng, Z., Zhang, B., Singhal, A., Du, D., McKinnon, D., Antropova, N., Bolukbasi, T., Keller, O., Reid, D., Finchelstein, D., Raad, M. A., Crocker, R., Hawkins, P., Dadashi, R., Gaffney, C., Franko, K., Bulanova, A., Leblond, R., Chung, S., Askham, H., Cobo, L. C., Xu, K., Fischer, F., Xu, J., Sorokin, C., Alberti, C., Lin, C.-C., Evans, C., Dimitriev, A., Forbes, H., Banarse, D., Tung, Z., Omernick, M., Bishop, C., Sterneck, R., Jain, R., Xia, J., Amid, E., Piccinno, F., Wang, X., Banzal, P., Mankowitz, D. J., Polozov, A., Krakovna, V., Brown, S., Bateni, M., Duan, D., Firoiu, V., Thotakuri, M., Natan, T., Geist, M., tan Girgin, S., Li, H., Ye, J., Roval, O., Tojo, R., Kwong, M., Lee-Thorp, J., Yew, C., Sinopalnikov, D., Ramos, S., Mellor, J., Sharma, A., Wu, K., Miller, D., Sonnerat, N., Vnukov, D., Greig, R., Beattie, J., Caveness, E., Bai, L., Eisenschlos, J., Korchemniy, A., Tsai, T., Jasarevic, M., Kong, W., Dao, P., Zheng, Z., Liu, F., Yang, F., Zhu, R., Teh, T. H., Sanmiya,

J., Gladchenko, E., Trdin, N., Toyama, D., Rosen, E., Tavakkol, S., Xue, L., Elkind, C., Woodman, O., Carpenter, J., Papamakarios, G., Kemp, R., Kafle, S., Grunina, T., Sinha, R., Talbert, A., Wu, D., Owusu-Afriyie, D., Du, C., Thornton, C., Pont-Tuset, J., Narayana, P., Li, J., Fatehi, S., Wieting, J., Ajmeri, O., Uria, B., Ko, Y., Knight, L., Héliou, A., Niu, N., Gu, S., Pang, C., Li, Y., Levine, N., Stolovich, A., Santamaria-Fernandez, R., Goenka, S., Yustalim, W., Strudel, R., Elqursh, A., Deck, C., Lee, H., Li, Z., Levin, K., Hoffmann, R., Holtmann-Rice, D., Bachem, O., Arora, S., Koh, C., Yeganeh, S. H., Põder, S., Tariq, M., Sun, Y., Ionita, L., Seyedhosseini, M., Tafti, P., Liu, Z., Gulati, A., Liu, J., Ye, X., Chrzaszcz, B., Wang, L., Sethi, N., Li, T., Brown, B., Singh, S., Fan, W., Parisi, A., Stanton, J., Koverkathu, V., Choquette-Choo, C. A., Li, Y., Lu, T., Ittycheriah, A., Shroff, P., Varadarajan, M., Bahargam, S., Willoughby, R., Gaddy, D., Desjardins, G., Cornero, M., Robenek, B., Mittal, B., Albrecht, B., Shenoy, A., Moiseev, F., Jacobsson, H., Ghaffarkhah, A., Rivière, M., Walton, A., Crepy, C., Parrish, A., Zhou, Z., Farabet, C., Radebaugh, C., Srinivasan, P., van der Salm, C., Fidjeland, A., Scellato, S., Latorre-Chimoto, E., Klimczak-Plucińska, H., Bridson, D., de Cesare, D., Hudson, T., Mendolicchio, P., Walker, L., Morris, A., Mauger, M., Guseynov, A., Reid, A., Odoom, S., Loher, L., Cotruta, V., Yenugula, M., Grewe, D., Petrushkina, A., Duerig, T., Sanchez, A., Yadlowsky, S., Shen, A., Globerson, A., Webb, L., Dua, S., Li, D., Bhupatiraju, S., Hurt, D., Qureshi, H., Agarwal, A., Shani, T., Eyal, M., Khare, A., Belle, S. R., Wang, L., Tekur, C., Kale, M. S., Wei, J., Sang, R., Saeta, B., Liechty, T., Sun, Y., Zhao, Y., Lee, S., Nayak, P., Fritz, D., Vuyyuru, M. R., Aslanides, J., Vyas, N., Wicke, M., Ma, X., Eltyshev, E., Martin, N., Cate, H., Manyika, J., Amiri, K., Kim, Y., Xiong, X., Kang, K., Luisier, F., Tripuraneni, N., Madras, D., Guo, M., Waters, A., Wang, O., Ainslie, J., Baldridge, J., Zhang, H., Pruthi, G., Bauer, J., Yang, F., Mansour, R., Gelman, J., Xu, Y., Polovets, G., Liu, J., Cai, H., Chen, W., Sheng, X., Xue, E., Ozair, S., Angermueller, C., Li, X., Sinha, A., Wang, W., Wiesinger, J., Koukoumidis, E., Tian, Y., Iyer, A., Gurumurthy, M., Goldenson, M., Shah, P., Blake, M., Yu, H., Urbanowicz, A., Palomaki, J., Fernando, C., Durden, K., Mehta, H., Momchev, N., Rahimtoroghi, E., Georgaki, M., Raul, A., Ruder, S., Redshaw, M., Lee, J., Zhou, D., Jalan, K., Li, D., Hechtman, B., Schuh, P., Nasr, M., Milan, K., Mikulik, V., Franco, J., Green, T., Nguyen, N., Kelley, J., Mahendru, A., Hu, A., Howland, J., Vargas, B., Hui, J., Bansal, K., Rao, V., Ghiya, R., Wang, E., Ye, K., Sarr, J. M., Preston, M. M., Elish, M., Li, S., Kaku, A., Gupta, J., Pasupat, I., Juan, D.-C., Someswar, M., M., T., Chen, X., Amini, A., Fabrikant, A., Chu, E., Dong, X., Muthal, A., Buthpitiya, S., Jauhari, S., Hua, N., Khandelwal, U., Hitron, A., Ren, J., Rinaldi, L., Drath, S., Dabush, A., Jiang, N.-J., Godhia, H., Sachs, U., Chen, A., Fan, Y., Taitelbaum, H., Noga, H., Dai, Z., Wang, J., Liang, C., Hamer, J., Ferng, C.-S., Elkind, C., Atias, A., Lee, P., Listík, V., Carlen, M., van de Kerkhof, J., Pikus, M., Zaher, K., Müller, P., Zykova, S., Stefanec, R., Gatsko, V., Hirnschall, C., Sethi, A., Xu, X. F., Ahuja, C., Tsai, B., Stefanoiu, A., Feng, B., Dhandhania, K., Katyal, M., Gupta, A., Parulekar, A., Pitta, D., Zhao, J., Bhatia, V., Bhavnani, Y., Alhadlaq, O., Li, X., Danenberg, P., Tu, D., Pine, A., Filippova, V., Ghosh, A., Limonchik, B., Urala, B., Lanka, C. K., Clive, D., Sun, Y., Li, E., Wu, H., Hongtongsak, K., Li, I., Thakkar, K., Omarov, K., Majmundar, K., Alverson, M., Kucharski, M., Patel, M., Jain, M., Zabelin, M., Pelagatti, P., Kohli, R., Kumar, S., Kim, J., Sankar, S., Shah, V., Ramachandruni, L., Zeng, X., Bariach, B., Weidinger, L., Vu, T., Andreev, A., He, A., Hui, K., Kashem, S., Subramanya, A., Hsiao, S., Hassabis, D., Kavukcuoglu, K., Sadovsky, A., Le, Q., Strohman, T., Wu, Y., Petrov, S., Dean, J., and Vinyals, O. Gemini: A family of highly capable multimodal models, 2024. URL https://arxiv.org/abs/2312.11805.

Valmeekam, K., Marquez, M., Olmo, A., Sreedharan, S., and Kambhampati, S. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36:38975–38987, 2023.

Veličković, P., Badia, A. P., Budden, D., Pascanu, R., Banino, A., Dashevskiy, M., Hadsell, R., and Blundell, C. The CLRS algorithmic reasoning benchmark. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 22084–22102. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/velickovic22a.html.

Wang, H., Feng, S., He, T., Tan, Z., Han, X., and Tsvetkov, Y. Can language models solve graph problems in natural language? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=UDqHhbqYJV.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

White, C., Dooley, S., Roberts, M., Pal, A., Feuer, B., Jain, S., Shwartz-Ziv, R., Jain, N., Saifullah, K., Dey, S., Shubh-Agrawal, Sandha, S. S., Naidu, S. V., Hegde, C., LeCun, Y., Goldstein, T., Neiswanger, W., and Goldblum,

M. Livebench: A challenging, contamination-limited LLM benchmark. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sKYHBTAxVa.

Xu, C., Guan, S., Greene, D., Kechadi, M., et al. Benchmark data contamination of large language models: A survey. *arXiv preprint arXiv:2406.04244*, 2024a.

Xu, J., Lu, L., Peng, X., Pang, J., Ding, J., Yang, L., Song, H., Li, K., Sun, X., Zhang, S., et al. Data set and benchmark (medgpteval) to evaluate responses from large language models in medicine: evaluation development and validation. *JMIR Medical Informatics*, 12(1):e57674, 2024b.

Yao, B., Jiang, M., Bobinac, T., Yang, D., and Hu, J. Benchmarking machine translation with cultural awareness. *arXiv preprint arXiv:2305.14328*, 2023.

Ye, J., Wang, Y., Huang, Y., Chen, D., Zhang, Q., Moniz, N., Gao, T., Geyer, W., Huang, C., Chen, P.-Y., et al. Justice or prejudice? quantifying biases in llm-as-a-judge. *arXiv preprint arXiv:2410.02736*, 2024.

Zha, J., Fan, Y., Yang, X., Gao, C., and Chen, X. How to enable llm with 3d capacity? a survey of spatial reasoning in llm. *arXiv preprint arXiv:2504.05786*, 2025.

Zhang, H., Da, J., Lee, D., Robinson, V., Wu, C., Song, W., Zhao, T., Raja, P., Zhuang, C., Slack, D., et al. A careful examination of large language model performance on grade school arithmetic. *Advances in Neural Information Processing Systems*, 37:46819–46836, 2024a.

Zhang, Y., Huang, Y., Sun, Y., Liu, C., Zhao, Z., Fang, Z., Wang, Y., Chen, H., Yang, X., Wei, X., et al. Benchmarking trustworthiness of multimodal large language models: A comprehensive study. *arXiv preprint arXiv:2406.07057*, 2024b.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

Zhou, H., Nova, A., Larochelle, H., Courville, A., Neyshabur, B., and Sedghi, H. Teaching algorithmic reasoning via in-context learning, 2022. URL https://arxiv.org/abs/2211.09066.

Zhou, H., Bradley, A., Littwin, E., Razin, N., Saremi, O., Susskind, J. M., Bengio, S., and Nakkiran, P. What algorithms can transformers learn? a study in length generalization. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=AssIuHnmHX.

Zhou, K., Zhu, Y., Chen, Z., Chen, W., Zhao, W. X., Chen, X., Lin, Y., Wen, J.-R., and Han, J. Don't make your llm an evaluation benchmark cheater. *arXiv preprint arXiv:2311.01964*, 2023.

Zhou, X., Zhao, X., and Li, G. Llm-enhanced data management, 2024b. URL https://arxiv.org/abs/2402.02643.

A. Appendix

A.1. Additional related works on LLM benchmarking

Large language models (LLMs) have demonstrated remarkable performance across a wide range of applications, prompting growing interest in understanding their capabilities and limitations. Recent efforts have focused on systematically benchmarking LLMs on core natural language processing tasks, including language understanding (Wang et al., 2024; Feng et al., 2024), text generation (Maynez et al., 2023; Singh et al., 2024), reasoning (Feng et al., 2024; Valmeekam et al., 2023), and machine translation (Yao et al., 2023). Additionally, some benchmarks evaluate ethical dimensions such as robustness, bias, and trustworthiness (Siska et al., 2024; Luo et al., 2024; Koo et al., 2023; Li et al., 2024c; Zhang et al., 2024b; Liu et al., 2023c). Specialized benchmarks have also emerged in scientific and technical domains, including mathematics (White et al., 2025; Liu et al., 2024a), programming (White et al., 2025; Jain et al., 2025; Chen et al., 2021; Zheng et al., 2023), data analysis (Sui et al., 2024; Liu et al., 2023b; White et al., 2025; Tang et al., 2023; Jiang et al., 2023), and medicine (Li et al., 2024b; Liu et al., 2023a; Longwell et al., 2024; Xu et al., 2024b; Kanithi et al., 2024).

While several studies examine how LLMs convert unstructured inputs into tabular or relational formats (Tang et al., 2023; Jiang et al., 2023), our work explores a distinct question: How well can LLMs construct, manipulate, and reason about classic *data structures* such as stacks, trees, and graphs? To our knowledge, this is the first comprehensive benchmark targeting this capability, offering a conceptually different evaluation from prior work on structured data.

A.2. Details of data structures and operations

In this section, we list the data structures and the corresponding operations tested in Table 8. We then provide detailed descriptions of each data structure and explain how we specify their implementations to eliminate ambiguity.

Array An array contains a list of elements stored in contiguous memory. We test its access, deletion, reversal, and search operations. To remove ambiguity, we specify that the array is 0-indexed. For operations like deletion, if duplicates exist, we delete the first occurrence. The final state is a list of elements in the array.

Stack A stack is a linear data structure that follows a Last-In-First-Out (LIFO) order. We test compound operations consisting of random sequences of push and pop from the top of the stack. The final state is a list of remaining elements in the stack.

Queue A queue is a linear data structure that follows a First-In-First-Out (FIFO) order. We evaluate compound operations of enqueue to the back and dequeue from the front. The final state is a list of remaining elements in the queue.

LRU Cache An LRU (Least Recently Used) cache stores a fixed number of items and evicts the least recently accessed one when full. We evaluate this caching operation with a sequence of requests as input. The final state is a set of elements in the LRU cache.

Priority Queue A priority queue stores elements with integer priorities, allowing access to the highest-priority element. We test compound operations including insert, remove, raise key, and decrease key, using a Fibonacci heap. Ties are broken by insertion order. The final state is a level-order traversal of the Fibonacci heap forest, outputting (value,priority) pairs, with nodes at each level sorted by descending priority and ties broken by larger value first.

Hashmap A hashmap is a key-value structure supporting fast access, insertion, and deletion via hashed keys. We test compound insert and delete operations, specifying the hash function and using chaining for collision resolution. The final state is a list of key-value pairs per bucket, preserving insertion order within each chain.

Trie A trie is a tree-based data structure for storing strings, where each node represents a character and paths from the root to leaves represent complete words, where common prefixes are shared. When generating strings, we increase the likelihood of shared prefixes to ensure the resulting trie has meaningful structure. The final state is a pre-order traversal of the trie, where each node's children are visited in lexicographical order to ensure an unambiguous representation.

Suffix Tree A suffix tree is a compressed trie built from all suffixes of a string, where each edge can represent multiple characters and each path from the root corresponds to a substring. We test the construction of a suffix tree from a given

word, appending a terminal character "$" to ensure a unique structure. The final state is a pre-order traversal collecting edge labels, with child edges visited in lexicographical order and "$" taking priority.

Table 8: Summary of data structures and associated operations in `DSR-Bench`. Data structures marked with * are included in the `DSR-Bench-challenge` subset. All compound operations without explicit specification consist of (insert, delete).

Category	Data Structure	Description	Operation	Application
Linear	Array	Contiguous memory	Access, Delete, Insert, Reverse, Search	Data storage
Temporal	Stack	LIFO (Last-In, First-Out)	Compound (Push, Pop)	Syntax parsing
	Queue	FIFO (First-In, First-Out)	Compound	OS management
	LRU Cache	Least-recently-used	Cache (Evict, Add)	Web broswers
	Priority Queue*	Priority ordering	Compound	Job scheduling
Associative	Hashmap	Key-value storage	Compound	Large-scaled storage
	Trie*	Hierarchical mapping of strings	Compound	Autocomplete
	Suffix Tree*	Text indexing via suffixes	Compound	DNA pattern matching
	Skip List*	Probabilistic layers for fast search	Compound	Concurrent databases
Hierarchical	Binary Search Tree	Hierarchical storage	Pre/In/Post-Order Traversal, Insert, Remove, Compound	Computer networks
	Heap*	Complete binary tree with priority ordering	Heapify, Compound	Memory management
	Red-Black Tree*	Self-balanced tree	Construct, Compound	Database indexes
	B+ Tree*	Multi-way balanced tree	Compound	File systems
	K-D Tree*	Hierarchical, spatial partition	Construct, Compound	3D graphics
	K-D Heap*	Hierarchical, complete binary tree, high dimensional priority	Compound	GPU job scheduling
Network	Graph	Many-to-many relationships	Breadth-First Traversal, Depth-First Traversal	Social networks
	Disjoint Set Union*	Sets partition & union	Compound (Union, Find)	Physics simulation
	Geometric Graph*	Graph modeling spatial data	Construct	Public transportation
Hybrid	Bloom Filter*	Probabilistic set and hashmap	Compound	Spam detection
	Directed Acyclic Word Graph*	Graph and trie tree	Compound	Compilers

Skip List A skip list is a probabilistic data structure composed of multiple layers of linked lists, where higher layers allow "skipping" over elements for faster access. We test compound operations of insert and delete. Insertion begins at the bottom layer, with the element randomly promoted to higher levels; pointers are updated at each level to preserve the structure. To remove ambiguity, promotion probabilities are explicitly specified in the prompts. The final state is represented as a list of lists, each corresponding to a layer of the skip list.

Binary Search Tree (BST) A binary search tree is a hierarchical structure where each node has at most two children: the left holds smaller values, and the right holds larger ones. We test insert, remove, tree traversals (pre-order, in-order, post-order), depth computation, and compound insert-remove operations. Inputs are guaranteed to contain no duplicates to ensure unique outputs. The final state for traversal tasks is a list of elements in the specified order, while for insert, remove, and compound tasks, it includes both pre-order and post-order traversals.

Heap A heap is a complete binary tree that satisfies the min-heap property, where each parent node is less than or equal to its children. We test both heapify and compound insert-delete operations using an array-based heap. Comparisons follow min-heap ordering, with ties broken by preferring the left child. The final state is the array representation of the heap.

Red-Black (RB) Tree A red-black tree is a self-balancing binary search tree where each node is colored red or black and must satisfy specific balance rules: no two consecutive red nodes are allowed, and all root-to-leaf paths must have the same number of black nodes. We test both construction and compound (insert, delete) operations. The final state is a pre-order traversal of the nodes, represented as tuples (value, color).

B+ Tree A B+ tree is a multi-way search tree used in databases and filesystems, where values are stored in leaf nodes and internal nodes serve as routing indexes. Leaf nodes are linked for efficient range queries. We specify splitting and merging rules to ensure unambiguous, balanced updates during compound insert and delete operations. The final state is a pre-order traversal of nodes, with keys in each node sorted in ascending order.

K-D Tree A K-D (k-dimensional) tree recursively partitions space by alternating the splitting axis at each level. Each node represents a point and divides the space into two halves based on a chosen coordinate. It is commonly used for spatial indexing, range queries, and nearest neighbor search. We test the construction of K-D trees across different dimensionalities, specifying the axis splitting sequence and tie-breaking rules (e.g., median selection for even-sized splits) to ensure consistency. The final state is a pre-order traversal of the tree.

K-D Heap A K-D heap maintains heap order based on a k-dimensional priority with a comparison metric, enabling efficient access to extremal points in multidimensional datasets. We test compound operations of insert and delete across different dimensionalities. We specify an array-based heap implementation, with comparisons based on Euclidean distance and tie-breaking rules that prefer the left child in case of a tie. The final state is a list of vectors representing the contents of the min-heap.

Graph A graph is a collection of nodes connected by edges, which can be directed or undirected, and is used to model networks, dependencies, and paths. We define graphs using edge list statements and test both breadth-first and depth-first traversals from a given source node, visiting neighbors in ascending order. Node values are unique to ensure consistent outputs. The final state is the list of nodes visited during the traversal.

Disjoint Set Union (DSU) A disjoint set union maintains a partition of elements into disjoint subsets, supporting efficient merges and membership queries. Internally, it forms a forest where each node points to a representative root. We test two operations: a sequence of unions between subsets, followed by queries for each element's representative. To ensure consistency, we specify that lower-rank roots are always attached to higher-rank ones. The final state lists the representative root of each input element in its original order.

Geometric (Geom) Graph Geometric graphs are graphs with nodes embedded in geometric space, typically Euclidean, where edges are formed based on spatial relationships such as proximity. They are widely used in robotics, computer graphics, and sensor networks where spatial structure is essential. We compute the Euclidean distance between each pair of points and add an edge if the distance is below a given threshold, assigning the edge a weight equal to that distance. The final state is a breadth-first traversal from a specified source node, exploring all neighbors at each level before proceeding. We specify the order of search based on the edge weights.

Bloom Filter A (counting) Bloom filter is a compact, probabilistic data structure for set membership testing, guaranteeing no false negatives and allowing a tunable false positive rate. It uses multiple hash functions to map each element to several positions in a counter array, incrementing or decrementing counts. We test on compound operations of insert and delete. We specify the hash functions used in the prompt to avoid ambiguity. The final state is the array of counters representing the Bloom filter.

Directed Acyclic Word Graph (DAWG) A Directed Acyclic Word Graph (DAWG) is a compressed data structure for storing a set of words, sharing both prefixes and suffixes. Nodes indicate whether they mark the end of a word, and edges are labeled with characters. Unlike a trie, a DAWG merges equivalent subtrees to reduce redundancy, making it well-suited for large static dictionaries and lexicon lookups. We test compound operations of insert and delete, specifying that merging

should occur at the final step along with the merging rules. To ensure a meaningful structure, we increase the likelihood of generating words with shared prefixes. The final state is a breadth-first traversal from the root (an empty string), where each node is recorded by the prefix it represents and whether it marks the end of a word.

A.3. Examples of prompting strategies

In this section, we illustrate prompting methods using compound operations of QUEUE as an example.

Stepwise This method expliclty adds a `step` attribute in the structured output, guiding the model to produce operations in a sequential and interpretable manner.

***Stepwise* prompting on compound operations of QUEUE.**

A queue is a data structure in which items are added at one end and removed from the other, maintaining a first-in, first-out (FIFO) order. You should create a queue. There are two types of operations: 1. (enqueue, k) means an element k is appended to the queue as the last element. 2. (dequeue) means the first element of the queue is deleted. You are given an empty queue initially.
Q: What is the final queue, when performing the following operations:

- (enqueue, 49)
- (dequeue)
- (enqueue, 86)
- (enqueue, 52)

Answer the question in 8000 tokens.

0-CoT This method appends the phrase "Let's think step by step" to the prompt to encourage reasoning without providing exemplars.

***0-CoT* prompting on compound operations of QUEUE.**

A queue is a data structure in which items are added at one end and removed from the other, maintaining a first-in, first-out (FIFO) order. You should create a queue. There are two types of operations: 1. (enqueue, k) means an element k is appended to the queue as the last element. 2. (dequeue) means the first element of the queue is deleted.

You are given an empty queue initially.

Q: What is the final queue, when performing the following operations:

- (enqueue, 49)
- (dequeue)
- (enqueue, 86)
- (enqueue, 52)

Let's think step by step. Answer the question in 8000 tokens.

CoT This strategy provides a single example that includes both intermediate reasoning steps and the final answer.

CoT prompting on compound operations of QUEUE.

A queue is a data structure in which items are added at one end and removed from the other, maintaining a first-in, first-out (FIFO) order. You should create a queue. There are two types of operations: 1. (enqueue, k) means an element k is appended to the queue as the last element. 2. (dequeue) means the first element of the queue is deleted. You are given an empty queue initially.
Q: What is the final queue, when performing the following operations:

- (enqueue, 21)
- (enqueue, 3)
- (dequeue)
- (dequeue)
- (enqueue, 48)

A: Initially, the queue is []. After (enqueue, 21), it becomes [21]. After (enqueue, 3), it becomes [21, 3]. After (dequeue), it becomes [3]. After (dequeue), it becomes []. After (enqueue, 48), it becomes [48]. The final queue is [48].

Q: What is the final queue, when performing the following operations:

- (enqueue, 49)
- (dequeue)
- (enqueue, 86)
- (enqueue, 52)

Answer the question in 8000 tokens.

3-shot This strategy provides three input-output examples to guide the model through pattern matching and demonstration.

3-shot prompting on compound operations of QUEUE.

A queue is a data structure in which items are added at one end and removed from the other, maintaining a first-in, first-out (FIFO) order. You should create a queue. There are two types of operations: 1. (enqueue, k) means an element k is appended to the queue as the last element. 2. (dequeue) means the first element of the queue is deleted. You are given an empty queue initially.
Q: What is the final queue, when performing the following operations:

- (enqueue, 21)
- (enqueue, 3)
- (dequeue)
- (dequeue)
- (enqueue, 48)

A: The final queue is [48]. (... Example 2...) (... Example 3...)

Q: What is the final queue, when performing the following operations:

- (enqueue, 49)
- (dequeue)
- (enqueue, 86)
- (enqueue, 52)

Answer the question in 8000 tokens.

None This method adds the instruction "No additional text needed" to prompt concise, direct answers that fit within the token limit and conform to the structured output format.

none prompting on compound operations of QUEUE.

A queue is a data structure in which items are added at one end and removed from the other, maintaining a first-in, first-out (FIFO) order. You should create a queue. There are two types of operations: 1. (enqueue, k) means an element k is appended to the queue as the last element. 2. (dequeue) means the first element of the queue is deleted. You are given an empty queue initially.
Q: What is the final queue, when performing the following operations:

- (enqueue, 49)
- (dequeue)
- (enqueue, 86)
- (enqueue, 52)

No additional text needed. Answer the question in 8000 tokens.

A.4. Accuracy by task and length level across all models

In this section, we provide supplementary accuracy tables for all models in `DSR-Bench`, broken down by task and length level. Table 9 summarizes the accuracy of instruction-tuned models on a subset of basic data structures across different length levels. Table 10 presents the accuracy of reasoning models on selected data structures from the `DSR-Bench-challenge` suite. For detailed per-model results across all tasks and length levels, see Table 11 (o4-mini), Table 12 (Gemini-2.5-Pro), Table 13 (Claude-3.7-Sonnet), Table 14 (DeepSeek-R1), Table 15 (GPT-4.1), Table 16 (Gemini-2.0-Flash), Table 17 (Claude-3.5-Sonnet), Table 18 (DeepSeek-V3), and Table 19 (Llama-3.3).

Table 9: Average accuracy on basic data structure tasks for instruction-tuned models (3 runs, scaled to [0, 1], rounded to two decimals).

Category	DS	Length	GPT-4.1	Gemini-2.0-Flash	Claude-3.5-Sonnet	DeepSeek-V3	Llama-3.3
Linear	Array	Short	0.98	0.98	1.00	1.00	0.89
		Medium	0.95	0.92	0.96	0.97	0.70
		Long	0.88	0.88	0.91	0.96	0.48
Temporal	Queue	Short	0.97	0.67	1.00	0.84	0.58
		Medium	0.49	0.37	1.00	0.38	0.12
		Long	0.18	0.03	0.98	0.07	0.06
	Stack	Short	0.97	0.67	1.00	0.70	0.09
		Medium	0.49	0.37	1.00	0.49	0.04
		Long	0.18	0.03	0.98	0.04	0.00
Associative	Hashmap	Short	0.19	0.28	0.37	0.00	0.00
		Medium	0.00	0.01	0.10	0.00	0.00
		Long	0.00	0.00	0.00	0.00	0.00
Hierarchical	BST	Short	0.82	0.63	0.89	0.76	0.46
		Medium	0.56	0.37	0.66	0.55	0.31
		Long	0.39	0.29	0.57	0.43	0.26
Network	Graph	Short	0.41	0.15	0.15	0.16	0.06
		Medium	0.05	0.02	0.02	0.02	0.01
		Long	0.00	0.00	0.00	0.00	0.00

Table 10: Average accuracy on the `DSR-Bench-challenge` suite for reasoning models (3 runs, scaled to [0, 1], rounded to two decimals).

Category	Data Structure	Length	o4-mini	Gemini-2.5-Pro	Claude-3.7-Sonnet	DeepSeek-R1
Temporal	Priority Queue	Short	0.89	0.87	0.70	0.92
		Medium	0.47	0.43	0.11	0.54
		Long	0.30	0.24	0.04	0.48
Associative	Trie	Short	0.99	1.00	0.23	0.93
		Medium	0.73	0.82	0.00	0.50
		Long	0.32	0.59	0.00	0.05
	Suffix Tree	Short	0.96	0.74	0.96	1.00
		Medium	0.87	0.92	0.98	0.98
		Long	0.37	0.89	0.79	0.90
	Skip List	Short	0.84	0.93	0.87	0.89
		Medium	0.60	0.64	0.76	0.63
		Long	0.41	0.61	0.61	0.54
Hierarchical	Heap	Short	0.61	0.67	0.70	0.63
		Medium	0.72	0.55	0.37	0.53
		Long	0.71	0.62	0.13	0.27
	Red Black Tree	Short	0.92	0.97	0.69	0.86
		Medium	0.67	0.73	0.11	0.63
		Long	0.37	0.60	0.12	0.37
	B+ Tree	Short	0.99	0.97	0.70	0.49
		Medium	0.98	0.93	0.32	0.23
		Long	0.94	0.96	0.13	0.21
	K-D Heap	Short	0.22	0.07	0.11	0.23
		Medium	0.09	0.08	0.03	0.08
		Long	0.00	0.00	0.00	0.01
	K-D Tree	Short	0.59	0.64	0.00	1.00
		Medium	0.43	0.81	0.00	0.34
		Long	0.38	0.64	0.00	0.01
Network	DSU	Short	1.00	1.00	1.00	1.00
		Medium	1.00	1.00	1.00	1.00
		Long	0.94	0.99	0.98	0.99
Hybrid	Bloom Filter	Short	1.00	0.99	0.44	0.99
		Medium	0.77	1.00	0.03	0.92
		Long	0.07	0.69	0.00	0.31
	DAWG	Short	0.49	0.07	0.17	0.40
		Medium	0.20	0.14	0.00	0.02
		Long	0.06	0.00	0.00	0.00
	Geom Graph	Short	0.37	0.30	0.07	0.36
		Medium	0.10	0.10	0.00	0.01
		Long	0.00	0.00	0.00	0.01

A.4.1. PERFORMANCE OF O4-MINI

Table 11: Mean (± std) accuracy of **o4-mini** on all `DSR-Bench` tasks over three runs. Data structures marked with * are included in the `DSR-Bench-challenge` suite.

Category	Data Structure	Operation	Short	Medium	Long
Linear	Array	Access	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Delete	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Insert	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Reverse	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Search	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Temporal	Stack	Compound	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Queue	Compound	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	LRU Cache	Cache	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Priority Queue*	Compound	0.89 (0.02)	0.47 (0.06)	0.30 (0.03)
Associative	Hashmap	Compound	0.89 (0.04)	0.37 (0.00)	0.26 (0.08)
	Trie*	Compound	0.99 (0.02)	0.73 (0.06)	0.32 (0.05)
	Suffix Tree*	Construct	0.96 (0.02)	0.87 (0.03)	0.37 (0.07)
	Skip List*	Compound	0.84 (0.02)	0.60 (0.03)	0.41 (0.02)
Associative	BST	Insert	1.00 (0.00)	1.00 (0.00)	0.99 (0.02)
		Remove	1.00 (0.00)	1.00 (0.00)	0.97 (0.03)
		In-Order Traversal	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Pre-Order Traversal	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Post-Order Traversal	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Depth	0.01 (0.02)	0.03 (0.03)	0.00 (0.00)
	Heap*	Compound	1.00 (0.00)	1.00 (0.00)	0.98 (0.02)
		Compound	0.77 (0.06)	0.86 (0.07)	0.74 (0.05)
		Heapify	0.44 (0.05)	0.58 (0.04)	0.67 (0.03)
	RB Tree*	Construct	0.90 (0.03)	0.32 (0.13)	0.05 (0.02)
		Compound	0.97 (0.00)	0.64 (0.04)	0.26 (0.02)
	B^+ Tree*	Compound	0.99 (0.02)	0.98 (0.04)	0.94 (0.02)
	K-D Tree*	Construct	0.59 (0.07)	0.43 (0.06)	0.38 (0.10)
	K-D Heap*	Compound	0.22 (0.02)	0.09 (0.02)	0.00 (0.00)
Network	Graph	Breadth-First Traversal	0.99 (0.02)	0.97 (0.03)	0.72 (0.14)
		Depth-First Traversal	1.00 (0.00)	0.88 (0.02)	0.64 (0.10)
	DSU*	Compound	1.00 (0.00)	1.00 (0.00)	0.94 (0.04)
	Geom Graph*	Construct	0.37 (0.00)	0.10 (0.00)	0.00 (0.00)
Network	Bloom Filter*	Compound	1.00 (0.00)	0.77 (0.09)	0.07 (0.00)
	DAWG*	Compound	0.49 (0.10)	0.20 (0.09)	0.06 (0.05)

A.4.2. PERFORMANCE OF GEMINI-2.5-PRO

Table 12: Mean (± std) accuracy of **Gemini-2.5-Pro** on all `DSR-Bench` tasks over three runs. Data structures marked with * are included in the `DSR-Bench-challenge` suite.

Category	Data Structure	Operation	Short	Medium	Long
Linear	Array	Access	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Delete	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Insert	1.00 (0.00)	0.98 (0.02)	1.00 (0.00)
		Reverse	1.00 (0.00)	0.98 (0.02)	1.00 (0.00)
		Search	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Temporal	Stack	Compound	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Queue	Compound	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	LRU Cache	Cache	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Priority Queue*	Compound	0.87 (0.03)	0.43 (0.03)	0.24 (0.05)
Associative	Hashmap	Compound	0.90 (0.06)	0.51 (0.04)	0.50 (0.07)
	Trie*	Compound	1.00 (0.00)	0.82 (0.05)	0.59 (0.08)
	Suffix Tree*	Construct	0.74 (0.08)	0.92 (0.08)	0.89 (0.02)
	Skip List*	Compound	0.93 (0.03)	0.64 (0.05)	0.61 (0.02)
Associative	BST	Insert	0.99 (0.02)	0.94 (0.05)	0.96 (0.02)
		Remove	0.99 (0.02)	0.84 (0.05)	0.90 (0.03)
		In-Order Traversal	1.00 (0.00)	1.00 (0.00)	0.99 (0.02)
		Pre-Order Traversal	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Post-Order Traversal	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Depth	0.22 (0.02)	0.17 (0.07)	0.18 (0.04)
	Heap*	Compound	1.00 (0.00)	1.00 (0.00)	0.99 (0.02)
		Compound	0.91 (0.04)	0.72 (0.11)	0.71 (0.05)
		Heapify	0.43 (0.21)	0.37 (0.03)	0.53 (0.09)
	RB Tree*	Construct	0.83 (0.05)	0.60 (0.03)	0.28 (0.07)
		Compound	0.91 (0.02)	0.47 (0.06)	0.18 (0.08)
	B^+ Tree*	Compound	0.97 (0.06)	0.93 (0.03)	0.96 (0.05)
	K-D Tree*	Construct	0.64 (0.11)	0.81 (0.05)	0.67 (0.12)
	K-D Heap*	Compound	0.07 (0.00)	0.08 (0.02)	0.00 (0.00)
Network	Graph	Breadth-First Traversal	0.98 (0.04)	0.99 (0.02)	0.86 (0.12)
		Depth-First Traversal	0.98 (0.02)	0.94 (0.04)	0.86 (0.05)
	DSU*	Compound	1.00 (0.00)	1.00 (0.00)	0.99 (0.02)
	Geom Graph*	Construct	0.30 (0.03)	0.10 (0.00)	0.00 (0.00)
Network	Bloom Filter*	Compound	0.99 (0.02)	1.00 (0.00)	0.69 (0.02)
	DAWG*	Compound	0.07 (0.02)	0.14 (0.08)	0.00 (0.00)

A.4.3. PERFORMANCE OF CLAUDE-3.7-SONNET

Table 13: Mean (± std) accuracy of **Claude-3.7-Sonnet** on all `DSR-Bench` tasks over three runs. Data structures marked with * are included in the `DSR-Bench-challenge` suite.

Category	Data Structure	Operation	Short	Medium	Long
Linear	Array	Access	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Delete	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Insert	1.00 (0.00)	1.00 (0.00)	0.96 (0.02)
		Reverse	1.00 (0.00)	0.98 (0.02)	0.97 (0.00)
		Search	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Temporal	Stack	Compound	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Queue	Compound	1.00 (0.00)	0.93 (0.00)	0.98 (0.02)
	LRU Cache	Compound	1.00 (0.00)	1.00 (0.00)	0.98 (0.02)
	Priority Queue*	Compound	0.70 (0.06)	0.11 (0.02)	0.04 (0.02)
Associative	Hashmap	Compound	0.71 (0.02)	0.16 (0.05)	0.04 (0.05)
	Trie*	Compound	0.94 (0.02)	0.64 (0.07)	0.31 (0.10)
	Suffix Tree*	Construct	0.23 (0.00)	0.00 (0.00)	0.00 (0.00)
	Skip List*	Compound	0.87 (0.06)	0.76 (0.05)	0.61 (0.11)
Associative	BST	Insert	0.96 (0.04)	0.98 (0.04)	0.79 (0.02)
		Remove	0.94 (0.02)	0.91 (0.02)	0.92 (0.02)
		In-Order Traversal	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Pre-Order Traversal	1.00 (0.00)	0.99 (0.02)	1.00 (0.00)
		Post-Order Traversal	1.00 (0.00)	0.74 (0.02)	0.93 (0.03)
		Depth	0.24 (0.11)	0.13 (0.03)	0.02 (0.04)
	Heap*	Compound	0.90 (0.03)	0.21 (0.02)	0.24 (0.05)
		Compound	0.70 (0.03)	0.32 (0.05)	0.13 (0.03)
		Heapify	0.89 (0.02)	0.62 (0.08)	0.26 (0.02)
	RB Tree*	Construct	0.19 (0.05)	0.00 (0.00)	0.00 (0.00)
		Compound	0.57 (0.03)	0.03 (0.00)	0.00 (0.00)
	B^+ Tree*	Compound	0.80 (0.00)	0.18 (0.02)	0.23 (0.03)
	K-D Tree*	Construct	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	K-D Heap*	Compound	0.11 (0.02)	0.03 (0.00)	0.00 (0.00)
Network	Graph	Breadth-First Traversal	0.40 (0.03)	0.08 (0.02)	0.01 (0.02)
		Depth-First Traversal	0.50 (0.03)	0.11 (0.02)	0.00 (0.00)
	DSU*	Compound	0.04 (0.02)	0.12 (0.04)	0.00 (0.00)
	Geom Graph*	Construct	0.07 (0.00)	0.00 (0.00)	0.00 (0.00)
Network	Bloom Filter*	Compound	0.44 (0.04)	0.03 (0.00)	0.00 (0.00)
	DAWG*	Compound	0.17 (0.00)	0.00 (0.00)	0.00 (0.00)

A.4.4. PERFORMANCE OF DEEPSEEK-R1

Table 14: Mean (± std) accuracy of **DeepSeek-R1** on all `DSR-Bench` tasks over three runs. Data structures marked with * are included in the `DSR-Bench-challenge` suite.

Category	Data Structure	Operation	Short	Medium	Long
Linear	Array	Access	1.00 (0.00)	0.98 (0.02)	1.00 (0.00)
		Delete	0.99 (0.02)	1.00 (0.00)	0.98 (0.02)
		Insert	0.98 (0.02)	0.99 (0.02)	0.99 (0.02)
		Reverse	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Search	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Temporal	Stack	Compound	1.00 (0.00)	1.00 (0.00)	0.94 (0.07)
	Queue	Compound	1.00 (0.00)	1.00 (0.00)	0.97 (0.00)
	LRU Cache	Compound	1.00 (0.00)	1.00 (0.00)	0.99 (0.01)
	Priority Queue*	Compound	0.92 (0.02)	0.54 (0.07)	0.48 (0.05)
Associative	Hashmap	Compound	0.44 (0.05)	0.01 (0.02)	0.03 (0.03)
	Trie*	Compound	0.54 (0.24)	0.32 (0.04)	0.12 (0.06)
	Suffix Tree*	Construct	0.93 (0.07)	0.50 (0.07)	0.05 (0.05)
	Skip List*	Compound	0.89 (0.04)	0.63 (0.03)	0.54 (0.02)
Associative	BST	Insert	1.00 (0.00)	0.98 (0.02)	0.90 (0.03)
		Remove	0.98 (0.02)	0.93 (0.03)	0.88 (0.05)
		In-Order Traversal	1.00 (0.00)	0.99 (0.02)	1.00 (0.00)
		Pre-Order Traversal	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Post-Order Traversal	1.00 (0.00)	1.00 (0.00)	0.97 (0.00)
		Depth	0.02 (0.04)	0.06 (0.04)	0.06 (0.05)
		Compound	0.97 (0.03)	0.84 (0.08)	0.65 (0.07)
	Heap*	Compound	0.49 (0.04)	0.23 (0.07)	0.21 (0.05)
		Heapify	0.34 (0.02)	0.16 (0.08)	0.08 (0.06)
	RB Tree*	Construct	0.88 (0.02)	0.10 (0.06)	0.00 (0.00)
		Compound	0.91 (0.04)	0.37 (0.10)	0.03 (0.03)
	B^+ Tree*	Compound	0.81 (0.02)	0.88 (0.04)	0.70 (0.06)
	K-D Tree*	Construct	1.00 (0.00)	0.34 (0.05)	0.01 (0.02)
	K-D Heap*	Compound	0.23 (0.00)	0.08 (0.02)	0.01 (0.02)
Network	Graph	Breadth-First Traversal	0.92 (0.02)	0.90 (0.06)	0.46 (0.05)
		Depth-First Traversal	0.80 (0.09)	0.58 (0.04)	0.22 (0.02)
	DSU*	Compound	0.64 (0.56)	0.92 (0.04)	0.83 (0.07)
	Geom Graph*	Construct	0.36 (0.02)	0.01 (0.02)	0.01 (0.02)
Network	Bloom Filter*	Compound	0.99 (0.02)	0.92 (0.02)	0.31 (0.02)
	DAWG*	Compound	0.40 (0.12)	0.02 (0.02)	0.00 (0.00)

A.4.5. PERFORMANCE OF GPT-4.1

Table 15: Mean (± std) accuracy of **GPT-4.1** on all `DSR-Bench` tasks over three runs. Data structures marked with * are included in the `DSR-Bench-challenge` suite.

Category	Data Structure	Operation	Short	Medium	Long
Linear	Array	Access	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Delete	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Insert	0.91 (0.08)	0.79 (0.02)	0.54 (0.02)
		Reverse	0.98 (0.02)	0.97 (0.00)	0.86 (0.02)
		Search	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Temporal	Stack	Compound	0.97 (0.00)	0.49 (0.02)	0.18 (0.04)
	Queue	Compound	0.82 (0.04)	0.59 (0.04)	0.19 (0.07)
	LRU Cache	Cache	0.94 (0.02)	0.80 (0.00)	0.81 (0.02)
	Priority Queue*	Compound	0.63 (0.03)	0.10 (0.00)	0.03 (0.00)
Associative	Hashmap	Compound	0.19 (0.07)	0.00 (0.00)	0.00 (0.00)
	Trie*	Compound	0.39 (0.07)	0.13 (0.03)	0.01 (0.02)
	Suffix Tree*	Construct	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Skip List*	Compound	0.21 (0.02)	0.00 (0.00)	0.00 (0.00)
Associative	BST	Insert	0.79 (0.04)	0.50 (0.03)	0.14 (0.02)
		Remove	0.78 (0.04)	0.58 (0.02)	0.36 (0.04)
		In-Order Traversal	1.00 (0.00)	1.00 (0.00)	0.94 (0.02)
		Pre-Order Traversal	1.00 (0.00)	0.97 (0.00)	0.98 (0.02)
		Post-Order Traversal	0.82 (0.02)	0.51 (0.04)	0.23 (0.06)
		Depth	0.66 (0.05)	0.08 (0.05)	0.03 (0.00)
	Heap*	Compound	0.69 (0.02)	0.26 (0.02)	0.03 (0.00)
		Compound	0.58 (0.02)	0.01 (0.02)	0.00 (0.00)
		Heapify	0.57 (0.03)	0.04 (0.02)	0.00 (0.00)
	RB Tree*	Construct	0.12 (0.02)	0.00 (0.00)	0.00 (0.00)
		Compound	0.31 (0.04)	0.02 (0.02)	0.00 (0.00)
	B^+ Tree*	Compound	0.27 (0.00)	0.30 (0.00)	0.13 (0.00)
	K-D Tree*	Construct	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	K-D Heap*	Compound	0.10 (0.00)	0.03 (0.00)	0.00 (0.00)
Network	Graph	Breadth-First Traversal	0.31 (0.05)	0.09 (0.02)	0.00 (0.00)
		Depth-First Traversal	0.50 (0.03)	0.00 (0.00)	0.00 (0.00)
	DSU*	Compound	0.06 (0.02)	0.00 (0.00)	0.00 (0.00)
	Geom Graph*	Construct	0.20 (0.03)	0.00 (0.00)	0.00 (0.00)
Network	Bloom Filter*	Compound	0.10 (0.00)	0.03 (0.00)	0.00 (0.00)
	DAWG*	Compound	0.16 (0.02)	0.00 (0.00)	0.00 (0.00)

A.4.6. PERFORMANCE OF GEMINI-2.0-FLASH

Table 16: Mean (± std) accuracy of **Gemini-2.0-Flash** on all `DSR-Bench` tasks over three runs. Data structures marked with * are included in the `DSR-Bench-challenge` suite.

Category	Data Structure	Operation	Short	Medium	Long
Linear	Array	Access	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Delete	0.96 (0.02)	0.87 (0.03)	0.77 (0.03)
		Insert	0.99 (0.02)	0.96 (0.02)	1.00 (0.00)
		Reverse	0.96 (0.02)	0.78 (0.02)	0.64 (0.04)
		Search	1.00 (0.00)	1.00 (0.00)	0.97 (0.00)
Temporal	Stack	Compound	0.67 (0.00)	0.37 (0.00)	0.03 (0.00)
	Queue	Compound	0.87 (0.00)	0.33 (0.00)	0.10 (0.00)
	LRU Cache	Cache	0.93 (0.00)	0.86 (0.02)	0.56 (0.02)
	Priority Queue*	Compound	0.38 (0.02)	0.10 (0.00)	0.01 (0.02)
Associative	Hashmap	Compound	0.28 (0.05)	0.01 (0.02)	0.00 (0.00)
	Trie*	Compound	0.31 (0.02)	0.18 (0.02)	0.03 (0.00)
	Suffix Tree*	Construct	0.00 (0.00)	0.02 (0.02)	0.00 (0.00)
	Skip List*	Compound	0.16 (0.02)	0.00 (0.00)	0.03 (0.00)
Associative	BST	Insert	0.31 (0.02)	0.27 (0.03)	0.06 (0.04)
		Remove	0.63 (0.09)	0.33 (0.03)	0.13 (0.03)
		In-Order Traversal	0.87 (0.00)	0.66 (0.02)	0.71 (0.04)
		Pre-Order Traversal	1.00 (0.00)	1.00 (0.00)	0.93 (0.00)
		Post-Order Traversal	0.63 (0.00)	0.17 (0.00)	0.10 (0.00)
		Depth	0.44 (0.04)	0.03 (0.00)	0.00 (0.00)
	Heap*	Compound	0.51 (0.05)	0.12 (0.04)	0.10 (0.00)
		Compound	0.32 (0.05)	0.03 (0.00)	0.02 (0.02)
		Heapify	0.23 (0.06)	0.00 (0.00)	0.00 (0.00)
	RB Tree*	Construct	0.08 (0.02)	0.00 (0.00)	0.00 (0.00)
		Compound	0.43 (0.03)	0.07 (0.00)	0.00 (0.00)
	B^+ Tree*	Compound	0.17 (0.00)	0.13 (0.03)	0.06 (0.05)
	K-D Tree*	Construct	0.02 (0.02)	0.00 (0.00)	0.00 (0.00)
	K-D Heap*	Compound	0.10 (0.00)	0.03 (0.00)	0.00 (0.00)
Network	Graph	Breadth-First Traversal	0.10 (0.00)	0.03 (0.00)	0.00 (0.00)
		Depth-First Traversal	0.19 (0.02)	0.00 (0.00)	0.00 (0.00)
	DSU*	Compound	0.01 (0.02)	0.00 (0.00)	0.00 (0.00)
	Geom Graph*	Construct	0.06 (0.02)	0.00 (0.00)	0.00 (0.00)
Network	Bloom Filter*	Compound	0.10 (0.00)	0.03 (0.00)	0.00 (0.00)
	DAWG*	Compound	0.18 (0.02)	0.00 (0.00)	0.00 (0.00)

A.4.7. PERFORMANCE OF CLAUDE-3.5-SONNET

Table 17: Mean (± std) accuracy of **Claude-3.5-Sonnet** on all `DSR-Bench` tasks over three runs. Data structures marked with * are included in the `DSR-Bench-challenge` suite.

Category	Data Structure	Operation	Short	Medium	Long
Linear	Array	Access	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Delete	1.00 (0.00)	1.00 (0.00)	0.93 (0.00)
		Insert	1.00 (0.00)	0.90 (0.00)	0.90 (0.00)
		Reverse	1.00 (0.00)	0.88 (0.02)	0.72 (0.05)
		Search	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Temporal	Stack	Compound	1.00 (0.00)	1.00 (0.00)	0.98 (0.04)
	Queue	Compound	0.87 (0.00)	0.67 (0.03)	0.79 (0.02)
	LRU Cache	Cache	0.99 (0.02)	0.90 (0.07)	0.58 (0.13)
	Priority Queue*	Compound	0.63 (0.00)	0.27 (0.03)	0.09 (0.02)
Associative	Hashmap	Compound	0.37 (0.03)	0.10 (0.00)	0.00 (0.00)
	Trie*	Compound	0.89 (0.04)	0.50 (0.03)	0.07 (0.00)
	Suffix Tree*	Construct	0.21 (0.02)	0.03 (0.00)	0.00 (0.00)
	Skip List*	Compound	0.77 (0.06)	0.30 (0.07)	0.20 (0.07)
Associative	BST	Insert	0.80 (0.06)	0.50 (0.06)	0.51 (0.05)
		Remove	0.96 (0.04)	0.87 (0.00)	0.77 (0.00)
		In-Order Traversal	0.97 (0.03)	0.94 (0.02)	0.94 (0.02)
		Pre-Order Traversal	1.00 (0.00)	1.00 (0.00)	0.99 (0.02)
		Post-Order Traversal	1.00 (0.00)	0.69 (0.08)	0.54 (0.02)
		Depth	0.76 (0.07)	0.34 (0.10)	0.16 (0.13)
	Heap*	Compound	0.77 (0.07)	0.28 (0.05)	0.09 (0.02)
		Compound	0.78 (0.04)	0.13 (0.00)	0.11 (0.02)
		Heapify	0.53 (0.12)	0.08 (0.04)	0.00 (0.00)
	RB Tree*	Construct	0.13 (0.00)	0.00 (0.00)	0.00 (0.00)
		Compound	0.44 (0.02)	0.03 (0.00)	0.00 (0.00)
	B^+ Tree*	Compound	0.40 (0.00)	0.28 (0.08)	0.02 (0.02)
	K-D Tree*	Construct	0.09 (0.02)	0.00 (0.00)	0.00 (0.00)
	K-D Heap*	Compound	0.13 (0.00)	0.02 (0.02)	0.00 (0.00)
Network	Graph	Breadth-First Traversal	0.17 (0.03)	0.02 (0.02)	0.00 (0.00)
		Depth-First Traversal	0.13 (0.03)	0.02 (0.02)	0.00 (0.00)
	DSU*	Compound	0.07 (0.03)	0.00 (0.00)	0.00 (0.00)
	Geom Graph*	Construct	0.06 (0.04)	0.00 (0.00)	0.00 (0.00)
Network	Bloom Filter*	Compound	0.10 (0.00)	0.03 (0.00)	0.00 (0.00)
	DAWG*	Compound	0.20 (0.00)	0.00 (0.00)	0.00 (0.00)

A.4.8. PERFORMANCE OF DEEPSEEK-V3

Table 18: Mean (± std) accuracy of **DeepSeek-V3** on all `DSR-Bench` tasks over three runs. Data structures marked with * are included in the `DSR-Bench-challenge` suite.

Category	Data Structure	Operation	Short	Medium	Long
Linear	Array	Access	1.00 (0.00)	0.97 (0.00)	0.97 (0.00)
		Delete	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Insert	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Reverse	1.00 (0.00)	0.92 (0.02)	0.92 (0.02)
		Search	1.00 (0.00)	0.97 (0.00)	0.93 (0.00)
Temporal	Stack	Compound	0.70 (0.03)	0.49 (0.02)	0.04 (0.02)
	Queue	Compound	0.84 (0.02)	0.38 (0.02)	0.07 (0.03)
	LRU Cache	Compound	0.94 (0.02)	0.77 (0.06)	0.76 (0.02)
	Priority Queue*	Compound	0.53 (0.03)	0.06 (0.04)	0.00 (0.00)
Associative	Hashmap	Compound	0.04 (0.02)	0.00 (0.00)	0.00 (0.00)
	Trie*	Compound	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Suffix Tree*	Construct	0.06 (0.02)	0.00 (0.00)	0.00 (0.00)
	Skip List*	Compound	0.06 (0.02)	0.00 (0.00)	0.00 (0.00)
Associative	BST	Insert	0.93 (0.03)	0.62 (0.02)	0.46 (0.05)
		Remove	0.84 (0.04)	0.80 (0.03)	0.66 (0.02)
		In-Order Traversal	0.97 (0.00)	1.00 (0.00)	1.00 (0.00)
		Pre-Order Traversal	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
		Post-Order Traversal	0.82 (0.02)	0.53 (0.03)	0.20 (0.03)
		Depth	0.92 (0.02)	0.43 (0.03)	0.08 (0.04)
	Heap*	Compound	0.68 (0.02)	0.12 (0.04)	0.06 (0.02)
		Compound	0.23 (0.00)	0.00 (0.00)	0.00 (0.00)
		Heapify	0.59 (0.02)	0.06 (0.02)	0.00 (0.00)
	RB Tree*	Construct	0.09 (0.02)	0.00 (0.00)	0.00 (0.00)
		Compound	0.30 (0.03)	0.00 (0.00)	0.00 (0.00)
	B^+ Tree*	Compound	0.14 (0.05)	0.10 (0.03)	0.00 (0.00)
	K-D Tree*	Construct	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	K-D Heap*	Compound	0.07 (0.00)	0.03 (0.00)	0.00 (0.00)
Network	Graph	Breadth-First Traversal	0.29 (0.05)	0.04 (0.02)	0.00 (0.00)
		Depth-First Traversal	0.22 (0.02)	0.03 (0.00)	0.00 (0.00)
	DSU*	Compound	0.03 (0.00)	0.00 (0.00)	0.00 (0.00)
	Geom Graph*	Construct	0.04 (0.02)	0.00 (0.00)	0.00 (0.00)
Network	Bloom Filter*	Compound	0.10 (0.00)	0.03 (0.00)	0.00 (0.00)
	DAWG*	Compound	0.17 (0.00)	0.00 (0.00)	0.00 (0.00)

A.4.9. PERFORMANCE OF LLAMA-3.3

Table 19: Mean (± std) accuracy of **Llama-3.3** on all `DSR-Bench` tasks over three runs. Data structures marked with * are included in the `DSR-Bench-challenge` suite.

Category	Data Structure	Operation	Short	Medium	Long
Linear	Array	Access	1.00 (0.00)	0.56 (0.04)	0.38 (0.02)
		Delete	0.81 (0.08)	0.68 (0.04)	0.44 (0.05)
		Insert	0.76 (0.04)	0.78 (0.02)	0.29 (0.07)
		Reverse	0.91 (0.02)	0.56 (0.02)	0.34 (0.07)
		Search	0.97 (0.00)	0.90 (0.00)	0.93 (0.00)
Temporal	Stack	Compound	0.09 (0.02)	0.04 (0.05)	0.00 (0.00)
	Queue	Compound	0.58 (0.11)	0.12 (0.02)	0.06 (0.02)
	LRU Cache	Compound	0.74 (0.08)	0.44 (0.11)	0.31 (0.11)
	Priority Queue*	Compound	0.21 (0.05)	0.01 (0.02)	0.02 (0.02)
Associative	Hashmap	Compound	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Trie*	Compound	0.01 (0.02)	0.00 (0.00)	0.00 (0.00)
	Suffix Tree*	Construct	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Skip List*	Compound	0.03 (0.00)	0.00 (0.00)	0.00 (0.00)
Associative	BST	Insert	0.37 (0.00)	0.14 (0.02)	0.03 (0.00)
		Remove	0.49 (0.04)	0.30 (0.03)	0.14 (0.04)
		In-Order Traversal	0.60 (0.06)	0.61 (0.08)	0.61 (0.04)
		Pre-Order Traversal	0.86 (0.11)	0.81 (0.08)	0.78 (0.11)
		Post-Order Traversal	0.31 (0.04)	0.04 (0.02)	0.00 (0.00)
		Depth	0.33 (0.03)	0.23 (0.00)	0.28 (0.05)
	Heap*	Compound	0.26 (0.02)	0.01 (0.02)	0.00 (0.00)
		Compound	0.17 (0.03)	0.03 (0.00)	0.00 (0.00)
		Heapify	0.24 (0.05)	0.00 (0.00)	0.00 (0.00)
	RB Tree*	Construct	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
		Compound	0.31 (0.02)	0.00 (0.00)	0.00 (0.00)
	B^+ Tree*	Compound	0.02 (0.04)	0.00 (0.00)	0.00 (0.00)
	K-D Tree*	Construct	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	K-D Heap*	Compound	0.02 (0.02)	0.02 (0.02)	0.00 (0.00)
Network	Graph	Breadth-First Traversal	0.07 (0.06)	0.00 (0.00)	0.00 (0.00)
		Depth-First Traversal	0.04 (0.05)	0.01 (0.02)	0.00 (0.00)
	DSU*	Compound	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Geom Graph*	Construct	0.07 (0.03)	0.00 (0.00)	0.00 (0.00)
Network	Bloom Filter*	Compound	0.00 (0.00)	0.07 (0.00)	0.00 (0.00)
	DAWG*	Compound	0.02 (0.02)	0.00 (0.00)	0.00 (0.00)

A.5. Accuracy by prompting methods across instruction-tuned models

This section presents additional accuracy tables for tasks in `DSR-Bench`, evaluating each instruction-tuned model across five prompting methods: **Stepwise**, **0-CoT**, **CoT**, **3-shot**, and **None**. The results are shown in Table 20 (GPT-4.1), Table 21 (Gemini-2.0-Flash), Table 22 (Claude-3.5-Sonnet), Table 23 (DeepSeek-V3), and Table 24 (Llama-3.3).

Table 20: Mean (± std) accuracy of **GPT-4.1** across prompting methods over three runs.

Data structure	Task	Stepwise	0-CoT	CoT	3-shot	None
Array	Access	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Delete	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Insert	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.91 (0.08)
	Reverse	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.99 (0.02)	0.98 (0.02)
	Search	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Queue	Compound	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.82 (0.04)
Stack	Compound	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.97 (0.00)
LRU Cache	Cache	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.94 (0.02)
Priority Queue	Compound	0.94 (0.04)	0.99 (0.01)	0.91 (0.02)	0.94 (0.02)	0.63 (0.03)
Hashmap	Compound	0.96 (0.02)	0.99 (0.01)	1.00 (0.00)	1.00 (0.00)	0.19 (0.07)
Trie	Compound	0.82 (0.02)	0.98 (0.00)	0.77 (0.07)	0.68 (0.02)	0.39 (0.07)
Suffix Tree	Construct	0.49 (0.07)	0.87 (0.01)	0.69 (0.04)	0.28 (0.08)	0.00 (0.00)
Skip List	Compound	0.77 (0.03)	0.94 (0.01)	0.56 (0.10)	0.84 (0.02)	0.21 (0.02)
BST	Insert	0.99 (0.02)	1.00 (0.00)	0.97 (0.00)	0.99 (0.02)	0.79 (0.04)
	Remove	0.99 (0.02)	1.00 (0.00)	0.99 (0.02)	1.00 (0.00)	0.78 (0.04)
	In-Order Traversal	0.98 (0.02)	1.00 (0.00)	1.00 (0.00)	0.98 (0.02)	1.00 (0.00)
	Pre-Order Traversal	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Post-Order Traversal	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.82 (0.02)
	Depth	0.14 (0.12)	0.14 (0.12)	0.07 (0.00)	0.14 (0.04)	0.66 (0.05)
	Compound	1.00 (0.00)	1.00 (0.00)	0.98 (0.02)	1.00 (0.00)	0.69 (0.02)
Heap	Compound	0.77 (0.07)	0.96 (0.01)	0.87 (0.06)	0.78 (0.14)	0.58 (0.02)
	Heapify	0.99 (0.02)	1.00 (0.01)	0.96 (0.04)	0.93 (0.07)	0.57 (0.03)
RB Tree	Construct	0.40 (0.13)	0.91 (0.02)	0.40 (0.12)	0.38 (0.05)	0.12 (0.02)
	Compound	0.77 (0.03)	0.96 (0.01)	0.37 (0.12)	0.70 (0.07)	0.31 (0.04)
B+ Tree	Compound	0.71 (0.05)	0.93 (0.01)	0.77 (0.03)	0.60 (0.06)	0.27 (0.00)
Graph	Breadth-First Traversal	0.90 (0.03)	0.94 (0.02)	0.83 (0.00)	0.82 (0.07)	0.31 (0.05)
	Depth-First Traversal	0.86 (0.05)	0.92 (0.02)	0.83 (0.03)	0.80 (0.03)	0.50 (0.03)
DSU	Compound	0.67 (0.03)	0.93 (0.02)	0.67 (0.07)	0.62 (0.05)	0.06 (0.02)
Bloom Filter	Compound	0.36 (0.07)	0.91 (0.02)	0.26 (0.08)	0.42 (0.07)	0.10 (0.00)
DAWG	Compound	0.20 (0.00)	0.79 (0.01)	0.21 (0.02)	0.20 (0.00)	0.16 (0.02)

Table 21: Mean (± std) accuracy of **Gemini-2.0-Flash** across prompting methods over three runs.

Data structure	Task	Stepwise	0-CoT	CoT	3-shot	None
Array	Access	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Delete	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.96 (0.02)
	Insert	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.99 (0.02)
	Reverse	0.67 (0.00)	0.78 (0.19)	0.56 (0.19)	1.00 (0.00)	0.96 (0.02)
	Search	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Queue	Compound	0.91 (0.05)	0.93 (0.03)	0.94 (0.02)	0.94 (0.02)	0.87 (0.00)
Stack	Compound	0.93 (0.07)	0.92 (0.04)	0.73 (0.09)	0.97 (0.03)	0.67 (0.00)
LRU Cache	Cache	0.97 (0.00)	0.96 (0.04)	0.82 (0.04)	0.87 (0.06)	0.93 (0.00)
Priority Queue	Compound	0.53 (0.12)	0.62 (0.12)	0.53 (0.03)	0.72 (0.05)	0.38 (0.02)
Hashmap	Compound	0.42 (0.08)	0.56 (0.11)	0.67 (0.09)	0.63 (0.07)	0.28 (0.05)
Trie	Compound	0.26 (0.04)	0.27 (0.12)	0.19 (0.07)	0.33 (0.03)	0.31 (0.02)
Suffix Tree	Construct	0.13 (0.00)	0.11 (0.05)	0.22 (0.07)	0.18 (0.05)	0.00 (0.00)
Skip List	Compound	0.18 (0.04)	0.31 (0.08)	0.27 (0.03)	0.31 (0.08)	0.16 (0.02)
BST	Insert	0.62 (0.02)	0.58 (0.13)	0.66 (0.02)	0.64 (0.08)	0.31 (0.02)
	Remove	0.64 (0.12)	0.67 (0.07)	0.73 (0.03)	0.69 (0.08)	0.63 (0.09)
	In-Order Traversal	0.87 (0.03)	0.86 (0.02)	0.92 (0.02)	0.80 (0.03)	0.87 (0.00)
	Pre-Order Traversal	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Post-Order Traversal	0.86 (0.05)	0.78 (0.04)	0.86 (0.04)	0.84 (0.08)	0.63 (0.00)
	Depth	0.69 (0.13)	0.66 (0.13)	0.63 (0.09)	0.54 (0.10)	0.44 (0.04)
	Compound	0.66 (0.08)	0.74 (0.08)	0.80 (0.03)	0.69 (0.08)	0.51 (0.05)
Heap	Compound	0.40 (0.09)	0.37 (0.07)	0.44 (0.13)	0.39 (0.05)	0.32 (0.05)
	Heapify	0.51 (0.11)	0.61 (0.05)	0.43 (0.09)	0.54 (0.08)	0.23 (0.06)
RB Tree	Construct	0.08 (0.07)	0.07 (0.03)	0.04 (0.02)	0.03 (0.00)	0.08 (0.02)
	Compound	0.41 (0.08)	0.44 (0.02)	0.28 (0.04)	0.31 (0.02)	0.43 (0.03)
B+ Tree	Compound	0.33 (0.09)	0.39 (0.08)	0.47 (0.03)	0.50 (0.09)	0.17 (0.00)
Graph	Breadth-First Traversal	0.17 (0.03)	0.24 (0.10)	0.36 (0.07)	0.20 (0.03)	0.10 (0.00)
	Depth-First Traversal	0.20 (0.09)	0.27 (0.12)	0.30 (0.03)	0.08 (0.02)	0.19 (0.02)
DSU	Compound	0.29 (0.02)	0.20 (0.00)	0.17 (0.09)	0.12 (0.02)	0.01 (0.02)
Bloom Filter	Compound	0.33 (0.07)	0.20 (0.00)	0.12 (0.04)	0.29 (0.08)	0.10 (0.00)
DAWG	Compound	0.16 (0.02)	0.18 (0.02)	0.13 (0.00)	0.14 (0.02)	0.18 (0.02)

Table 22: Mean (± std) accuracy of **Claude-3.5-Sonnet** across prompting methods over three runs.

Data structure	Task	Stepwise	0-CoT	CoT	3-shot	None
Array	Access	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Delete	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Insert	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Reverse	1.00 (0.00)	0.99 (0.02)	0.96 (0.04)	0.99 (0.02)	1.00 (0.00)
	Search	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Queue	Compound	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.87 (0.00)
Stack	Compound	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
LRU Cache	Cache	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.99 (0.02)
Priority Queue	Compound	0.94 (0.02)	0.96 (0.02)	0.90 (0.03)	0.94 (0.02)	0.63 (0.00)
Hashmap	Compound	0.89 (0.02)	0.81 (0.04)	1.00 (0.00)	1.00 (0.00)	0.37 (0.03)
Trie	Compound	0.02 (0.02)	0.11 (0.07)	0.00 (0.00)	0.11 (0.02)	0.89 (0.04)
Suffix Tree	Construct	0.29 (0.08)	0.24 (0.07)	0.56 (0.08)	0.27 (0.03)	0.21 (0.02)
Skip List	Compound	0.82 (0.02)	0.77 (0.03)	0.64 (0.04)	0.61 (0.02)	0.77 (0.06)
BST	Insert	1.00 (0.00)	1.00 (0.00)	0.96 (0.04)	0.92 (0.05)	0.80 (0.06)
	Remove	1.00 (0.00)	1.00 (0.00)	0.97 (0.00)	0.98 (0.02)	0.96 (0.04)
	In-Order Traversal	0.88 (0.05)	0.97 (0.06)	1.00 (0.00)	0.98 (0.02)	0.97 (0.03)
	Pre-Order Traversal	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Post-Order Traversal	0.99 (0.02)	0.99 (0.02)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Depth	0.64 (0.02)	0.79 (0.11)	0.09 (0.05)	0.08 (0.04)	0.76 (0.07)
	Compound	0.89 (0.05)	0.88 (0.05)	0.88 (0.04)	0.86 (0.05)	0.77 (0.07)
Heap	Compound	0.69 (0.02)	0.69 (0.02)	0.66 (0.04)	0.68 (0.02)	0.78 (0.04)
	Heapify	0.73 (0.00)	0.69 (0.04)	0.33 (0.06)	0.80 (0.07)	0.53 (0.12)
RB Tree	Construct	0.11 (0.08)	0.08 (0.05)	0.19 (0.07)	0.21 (0.02)	0.13 (0.00)
	Compound	0.57 (0.00)	0.60 (0.03)	0.03 (0.00)	0.13 (0.03)	0.44 (0.02)
B+ Tree	Compound	0.61 (0.02)	0.69 (0.04)	0.67 (0.03)	0.44 (0.05)	0.40 (0.00)
Graph	Breadth-First Traversal	0.30 (0.09)	0.32 (0.04)	0.52 (0.02)	0.26 (0.04)	0.17 (0.03)
	Depth-First Traversal	0.30 (0.09)	0.24 (0.04)	0.26 (0.05)	0.23 (0.03)	0.13 (0.03)
DSU	Compound	0.53 (0.07)	0.49 (0.05)	0.76 (0.08)	0.53 (0.09)	0.07 (0.03)
Bloom Filter	Compound	0.12 (0.04)	0.12 (0.02)	0.10 (0.00)	0.10 (0.00)	0.10 (0.00)
DAWG	Compound	0.17 (0.06)	0.19 (0.02)	0.18 (0.02)	0.19 (0.02)	0.20 (0.00)

Table 23: Mean (± std) accuracy of **DeepSeek-V3** across prompting methods over three runs.

Data structure	Task	Stepwise	0-CoT	CoT	3-shot	None
Array	Access	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Delete	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Insert	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Reverse	1.00 (0.00)	1.00 (0.00)	0.99 (0.00)	1.00 (0.00)	1.00 (0.00)
	Search	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Queue	Compound	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.84 (0.02)
Stack	Compound	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.70 (0.03)
LRU Cache	Cache	0.93 (0.00)	0.97 (0.00)	0.98 (0.02)	0.90 (0.00)	0.94 (0.02)
Priority Queue	Compound	0.86 (0.04)	0.79 (0.05)	0.84 (0.02)	0.82 (0.02)	0.53 (0.03)
Hashmap	Compound	1.00 (0.00)	1.00 (0.00)	0.90 (0.03)	1.00 (0.00)	0.04 (0.02)
Trie	Compound	0.00 (0.00)	0.00 (0.00)	0.63 (0.00)	0.64 (0.02)	0.00 (0.00)
Suffix Tree	Construct	0.19 (0.02)	0.18 (0.02)	0.39 (0.04)	0.19 (0.02)	0.06 (0.02)
Skip List	Compound	0.08 (0.02)	0.06 (0.02)	0.19 (0.04)	0.08 (0.02)	0.06 (0.02)
BST	Insert	0.94 (0.02)	0.98 (0.02)	0.91 (0.02)	0.87 (0.06)	0.93 (0.03)
	Remove	0.92 (0.04)	0.70 (0.03)	0.84 (0.04)	0.82 (0.05)	0.84 (0.04)
	In-Order Traversal	0.94 (0.04)	0.92 (0.02)	0.94 (0.04)	0.93 (0.00)	0.97 (0.00)
	Pre-Order Traversal	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	Post-Order Traversal	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.99 (0.02)	0.82 (0.02)
	Depth	0.07 (0.03)	0.03 (0.03)	0.00 (0.00)	0.19 (0.04)	0.92 (0.02)
	Compound	0.81 (0.02)	0.77 (0.03)	0.96 (0.02)	0.71 (0.04)	0.68 (0.02)
RB Tree	Construct	0.04 (0.04)	0.02 (0.04)	0.06 (0.02)	0.07 (0.06)	0.09 (0.02)
	Compound	0.63 (0.03)	0.67 (0.03)	0.12 (0.02)	0.59 (0.08)	0.30 (0.03)
B+ Tree	Compound	0.71 (0.04)	0.66 (0.02)	0.44 (0.08)	0.38 (0.02)	0.14 (0.05)
Heap	Compound	0.83 (0.03)	0.87 (0.03)	0.87 (0.03)	0.89 (0.05)	0.23 (0.00)
	Heapify	0.83 (0.06)	0.83 (0.03)	0.57 (0.03)	0.81 (0.02)	0.59 (0.02)
Graph	Breadth-First Traversal	0.51 (0.02)	0.51 (0.04)	0.29 (0.08)	0.00 (0.00)	0.29 (0.05)
	Depth-First Traversal	0.23 (0.03)	0.39 (0.05)	0.36 (0.05)	0.00 (0.00)	0.22 (0.02)
DSU	Compound	0.34 (0.04)	0.41 (0.05)	0.30 (0.03)	0.16 (0.02)	0.03 (0.00)
Bloom Filter	Compound	0.10 (0.00)	0.10 (0.00)	0.10 (0.00)	0.10 (0.00)	0.10 (0.00)
DAWG	Compound	0.18 (0.02)	0.18 (0.02)	0.17 (0.00)	0.18 (0.02)	0.17 (0.00)

Table 24: Mean (± std) accuracy of **Llama-3.3** across prompting methods over three runs.

Data structure	Task	Stepwise	0-CoT	CoT	3-shot	None
Array	Access	0.98 (0.02)	1.00 (0.00)	0.97 (0.03)	0.99 (0.02)	0.98 (0.02)
	Delete	0.92 (0.05)	0.96 (0.02)	0.78 (0.05)	0.82 (0.08)	0.81 (0.08)
	Insert	0.78 (0.02)	0.88 (0.07)	0.81 (0.04)	0.89 (0.10)	0.76 (0.04)
	Reverse	0.96 (0.02)	0.87 (0.06)	0.50 (0.07)	0.74 (0.02)	0.91 (0.02)
	Search	0.99 (0.02)	0.99 (0.02)	0.96 (0.02)	0.98 (0.02)	1.00 (0.00)
Skip List	Compound	0.14 (0.04)	0.10 (0.03)	0.18 (0.02)	0.20 (0.07)	0.03 (0.00)
Queue	Compound	0.93 (0.03)	0.94 (0.05)	0.94 (0.02)	0.81 (0.05)	0.58 (0.11)
Stack	Compound	0.88 (0.02)	0.86 (0.13)	0.88 (0.10)	0.68 (0.07)	0.09 (0.02)
LRU Cache	Compound	0.89 (0.02)	0.92 (0.05)	0.98 (0.04)	0.82 (0.07)	0.74 (0.08)
Priority Queue	Compound	0.73 (0.03)	0.73 (0.10)	0.69 (0.02)	0.76 (0.04)	0.21 (0.05)
Hashmap	Compound	0.39 (0.16)	0.23 (0.07)	0.32 (0.04)	0.13 (0.00)	0.00 (0.00)
Trie	Compound	0.01 (0.02)	0.00 (0.00)	0.10 (0.09)	0.14 (0.10)	0.01 (0.02)
Suffix Tree	Construct	0.01 (0.02)	0.01 (0.02)	0.07 (0.03)	0.00 (0.00)	0.00 (0.00)
BST	Insert	0.53 (0.07)	0.52 (0.16)	0.38 (0.05)	0.37 (0.09)	0.37 (0.00)
	Remove	0.57 (0.03)	0.57 (0.15)	0.40 (0.25)	0.60 (0.06)	0.49 (0.04)
	In-Order Traversal	0.56 (0.08)	0.51 (0.13)	0.69 (0.02)	0.60 (0.12)	0.60 (0.06)
	Pre-Order Traversal	0.60 (0.15)	0.78 (0.10)	0.67 (0.03)	0.82 (0.07)	0.86 (0.11)
	Post-Order Traversal	0.54 (0.26)	0.61 (0.05)	0.90 (0.03)	0.71 (0.04)	0.31 (0.04)
	Depth	0.40 (0.15)	0.51 (0.04)	0.98 (0.02)	0.77 (0.07)	0.33 (0.03)
	Compound	0.54 (0.16)	0.51 (0.05)	0.50 (0.09)	0.49 (0.10)	0.27 (0.15)
Heap	Compound	0.33 (0.06)	0.33 (0.03)	0.32 (0.05)	0.41 (0.11)	0.17 (0.03)
	Heapify	0.29 (0.02)	0.23 (0.12)	0.08 (0.02)	0.18 (0.05)	0.24 (0.05)
RB Tree	Construct	0.01 (0.02)	0.01 (0.02)	0.01 (0.02)	0.07 (0.03)	0.00 (0.00)
	Compound	0.36 (0.04)	0.30 (0.10)	0.03 (0.00)	0.28 (0.07)	0.31 (0.02)
B+ Tree	Compound	0.17 (0.03)	0.18 (0.02)	0.50 (0.00)	0.23 (0.03)	0.02 (0.04)
Graph	Breadth-First Traversal	0.12 (0.02)	0.12 (0.07)	0.28 (0.08)	0.06 (0.07)	0.04 (0.05)
	Depth-First Traversal	0.12 (0.08)	0.14 (0.02)	0.09 (0.05)	0.10 (0.07)	0.07 (0.06)
DSU	Construct	0.08 (0.05)	0.04 (0.02)	0.37 (0.15)	0.07 (0.00)	0.00 (0.00)
Bloom Filter	Compound	0.01 (0.02)	0.01 (0.02)	0.00 (0.00)	0.01 (0.02)	0.00 (0.00)
DAWG	Compound	0.03 (0.03)	0.02 (0.02)	0.06 (0.02)	0.07 (0.06)	0.02 (0.02)

A.6. `DSR-Bench-spatial` supplementary materials

Figure 3 presents example illustrations of the non-uniform input distributions: circles, moons, and blobs, which were adopted from `scikit-learn`. These synthetic patterns are used to evaluate whether models can adapt to irregular and non-uniform spatial distributions, an essential aspect of real-world data, as discussed in Section 4.2.

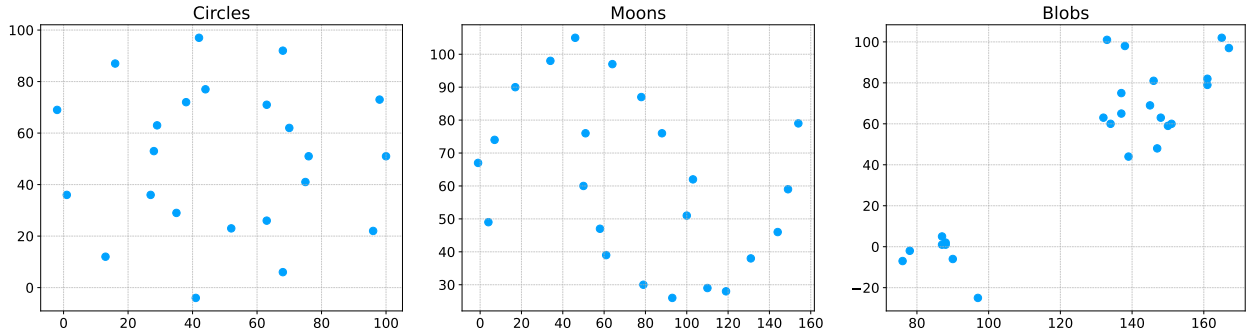

Figure 3: Example K-D Tree instances from three non-uniform distributions.

A.7. `DSR-Bench-natural` supplementary materials

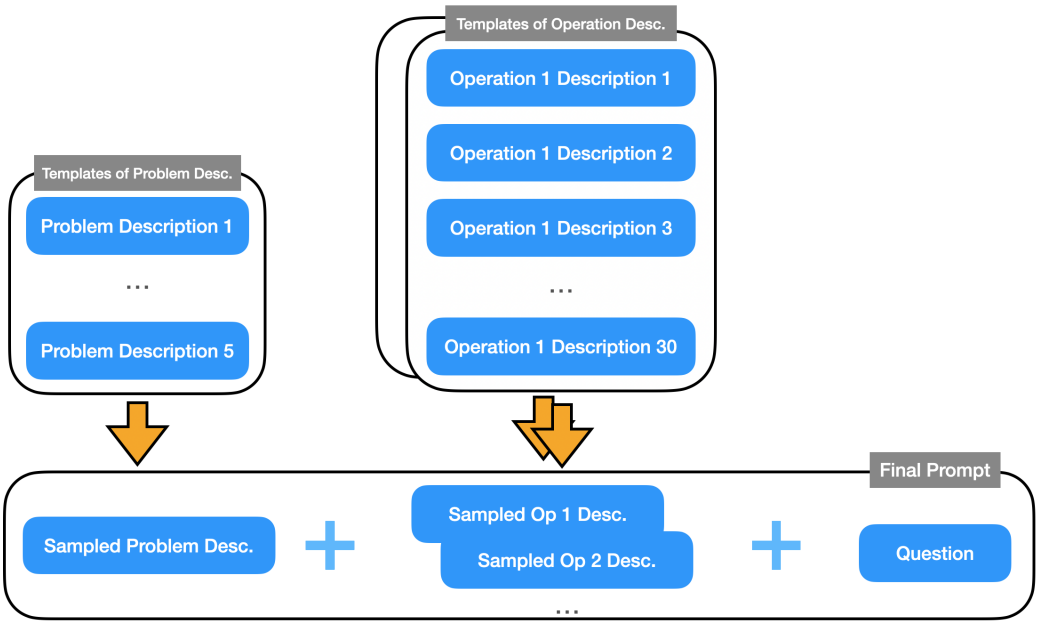

Figure 4: The pipeline for generating natural language prompts.

Figure 4 illustrates our generation process for natural language prompts in `DSR-Bench-natural`. For each data structure task, we begin by manually writing an initial scenario narrative (see On a sunny afternoon... arrived in the queue" in A.7). This narrative is then paraphrased into five variants using `GPT-4o`. For each supported operation type (e.g., enqueue, dequeue), we generate 30 paraphrased operation templates in a similar manner (e.g., With money in hand, Yuki Lopez stepped into the queue." corresponds to "enqueue Yuki Lopez" in A.7). In total, each task is backed by a pool of five scenario descriptions and 30 templates per operation type.

We use the same input distribution as in the `DSR-Bench`, replacing synthetic values (e.g. "enqueue 5") with realistic values (e.g. "enqueue Yuki Lopez"). During prompt construction, we randomly sample one scenario description. For each data structure operation, we instantiate it with a synthetically generated value (e.g., a name) and sample one operation template

to form a complete instance. All templates were reviewed by three human annotators to ensure clarity and unambiguous solvability.

Queue (children buying ice cream) We construct a real-world scenario that implicitly models a QUEUE: children lining up to buy ice cream from a truck. The enqueue operation corresponds to a child joining the end of the line, while the dequeue operation represents a child being served from the front. The scenario explicitly enforces the FIFO discipline by stating that no skipping is allowed.

Example prompt from the natural language extension for QUEUE.

On a sunny afternoon in the neighborhood park, an ice cream truck rolled in, its cheerful tune drawing children from all directions. The children began to form a line. Each child joined at the end while the vendor served at the front. Coins jingled in pockets while the children eagerly discussed the different flavors. The children were served in the order they had arrived in the queue.

- Fatima Singh ran over from the swings and joined the ice cream line.
- With money in hand, Yuki Lopez stepped into the queue.
- Haruto Sanchez spotted the growing line and quietly joined.
- A cone was handed over, and the line moved on.
- After hearing about the ice cream truck from Fatima Singh, Isabella Miller decided to line up too.
- One more excited customer walked off with a cone in hand.
- Carlos Martinez joined the queue after Yuki Lopez mentioned how good the ice cream looked.

Q: What is the order of the remaining kids in line? Your answer should be a list of names.
Answer the question in 8000 tokens.

BST (clinic appointments) We construct a scenario in which a clinic uses a BST to store patient appointments, ordered by appointment time and tie-broken by the patient's name. The insert operation adds a name, time) appointment to the tree, while the delete operation corresponds to a patient canceling their appointment. To retrieve all records, a pre-order traversal of the tree is performed.

Example prompt from the natural language extension for BST.

A local clinic uses an appointment management system which maintains a binary search tree to store appointments. Each appointment (name, appointment time) is a tuple of two strings, e.g. ('Alice Baker', '10:30'), and is represented by a node in the tree. Order is maintained by appointment time. The alphabetical order of the patient name is used to break ties. During data retrieval (i.e. to print out all of the appointments), a pre-order traversal is used, starting from the root node. Initially the tree is empty.

- Hassan Chen joined the list at 09:22 successfully.
- Knowing Hassan Chen has booked, Amelia Martinez was placed at 13:11.
- Hassan Chen hesitated a lot but still decided to cancel.
- As a friend Harper Young recommended, Lucas Fernandez quickly booked at 09:18.
- Harper Young was scheduled for 15:48, slightly earlier than 16:01.

Q: What is the pre-order traversal of the appointment schedule following the binary search tree? Your answer should be a list of (name, appointment time) in the format of a tuple of two strings. Answer the question in 8000 tokens.

Graph (galaxy traveling) We create a scenario set in a galaxy where planets are connected by space tunnels. The task is to navigate a starship to visit as many planets as possible using depth-first search, starting from a given planet and visiting neighbors in lexicographical order.

Example prompt from the natural language extension for GRAPH.

You pilot a Star Courier through a galaxy of planets. Your job is to travel to as many planets as possible via bidirectional space tunnels, starting from a source planet. The courier computes its route with depth-first search, and whenever multiple unvisited neighbors are available it selects the neighbor with the alphabetically earliest planet name.

- Star maps show a space tunnel running between Triton and Pulsar.
- A space tunnel links Ganymede and Wraith.
- The tunnel from Triton to Ganymede is well-known for its convenience.
- There's a tunnel between Fenrir and Vega.
- The tunnel linking Fenrir and Pulsar is a crucial route for all space dwellers.
- Long-range scans confirm a navigable tunnel between Triton and Orion.
- Though Vega is nearby, the space team decides to connect Nereus and Pulsar via a tunnel.
- Vega and Orion are part of the same local cluster, connected by a space tunnel.
- Vega and Ymir are directly linked by a tunnel monitored by the space police.
- The ancient network includes a direct tunnel between Wraith and Pulsar.
- Only Pulsar and Orion are reachable via this tunnel — not Nereus.

Q: What is the full DFS traversal order (as a list of planet names) starting from Fenrir? Answer the question in 8000 tokens.

A.8. Levenshtein distance: an auxiliary metric for `DSR-Bench`

In addition to the binary (0/1) accuracy reported in the main text, `DSR-Bench` includes an optional evaluation metric based on *Levenshtein distance*, which measures the minimum number of single-character insertions, deletions, or substitutions needed to transform one string into another. As a continuous metric, it captures degrees of error that binary accuracy flattens. For instance, given the correct output `[1, 3, 6]`, the prediction `[3, 1, 6]` is clearly closer than `[0, 0, 0]`, and Levenshtein distance reflects that nuance.

However, this granularity can also blur important semantic distinctions. A syntactically well-formed but semantically incorrect output may still receive a high score, especially when the expected answer is long or formatted. When averaged over 30 test cases, models with large gaps in binary accuracy can appear deceptively similar under Levenshtein. For example, the SKIP LIST and DSU compound tasks yield the same Levenshtein score (0.75), despite the former achieving more than triple the binary accuracy (Table 25).

Output length further complicates cross-task comparison. Tasks with short, single-token outputs (e.g., BINARY SEARCH TREE depth) tend to show similar binary and Levenshtein scores (e.g., 0.66), while longer, multi-token outputs (e.g., GRAPH BFS) inflate Levenshtein scores (0.82) even when binary accuracy remains low (0.31). Relying on Levenshtein distance alone may thus give a misleading impression—for example, that the model performs well on BFS but poorly on tree depth—when binary accuracy indicates the opposite.

For these reasons, we report all results using binary accuracy and relegate Levenshtein evaluation to the toolkit. The implementation remains publicly available, as the metric can still offer a useful secondary perspective, particularly when comparing models on the same task and output length.

Table 25: Mean (± std) binary accuracy vs. Levenshtein distance scores across tasks using GPT-4.1.

Data structure	Operation	Binary	Levenshtein
Array	Access	1.00 (0.00)	1.00 (0.00)
Queue	Compound	0.82 (0.04)	0.96 (0.01)
Stack	Compound	0.97 (0.00)	0.99 (0.00)
LRU Cache	Cache	0.94 (0.02)	1.00 (0.00)
Priority Queue	Compound	0.63 (0.03)	0.89 (0.01)
Hashmap	Compound	0.19 (0.07)	0.75 (0.02)
Trie	Compound	0.39 (0.07)	0.90 (0.02)
Suffix Tree	Construct	0.00 (0.00)	0.49 (0.01)
Skip List	Compound	0.21 (0.02)	0.75 (0.01)
BST	Insert	0.79 (0.04)	0.97 (0.01)
	Remove	0.78 (0.04)	0.95 (0.01)
	Post-Order Traversal	0.82 (0.02)	0.94 (0.01)
	Depth	0.66 (0.05)	0.66 (0.05)
	Compound	0.69 (0.02)	0.88 (0.01)
Heap	Compound	0.58 (0.02)	0.87 (0.01)
	Heapify	0.57 (0.03)	0.94 (0.01)
RB Tree	Construct	0.12 (0.02)	0.87 (0.00)
	Compound	0.31 (0.04)	0.88 (0.02)
B+ Tree	Compound	0.27 (0.00)	0.79 (0.00)
Graph	Breadth-First Traversal	0.31 (0.05)	0.82 (0.01)
	Depth-First Traversal	0.50 (0.03)	0.85 (0.01)
DSU	Compound	0.06 (0.02)	0.75 (0.01)
Bloom Filter	Compound	0.10 (0.00)	0.84 (0.01)
DAWG	Compound	0.16 (0.02)	0.74 (0.02)