LONGREASONARENA: A Long Reasoning Benchmark for Large Language Models

Anonymous ACL submission

Abstract

Existing long-context benchmarks for Large Language Models (LLMs) focus on evaluating comprehension of long inputs, while overlooking the evaluation of long reasoning abilities. To address this gap, we introduce LONGREASONARENA, a benchmark specifically designed to assess the long reasoning capabilities of LLMs. Our tasks require models to solve problems by executing multi-step algorithms that reflect key aspects of long reasoning, such as retrieval and backtracking. By controlling the inputs, the required reasoning length can be arbitrarily scaled, reaching up to 1 million tokens of reasoning for the most challenging tasks. Extensive evaluation results demonstrate that LONGREASONARENA presents a significant challenge for both open-source and proprietary LLMs. For instance, Deepseek-R1 achieves only 7.5% accuracy on our task. Further analysis also reveals that the accuracy exhibits a linear decline with respect to the logarithm of the expected number of reasoning steps.

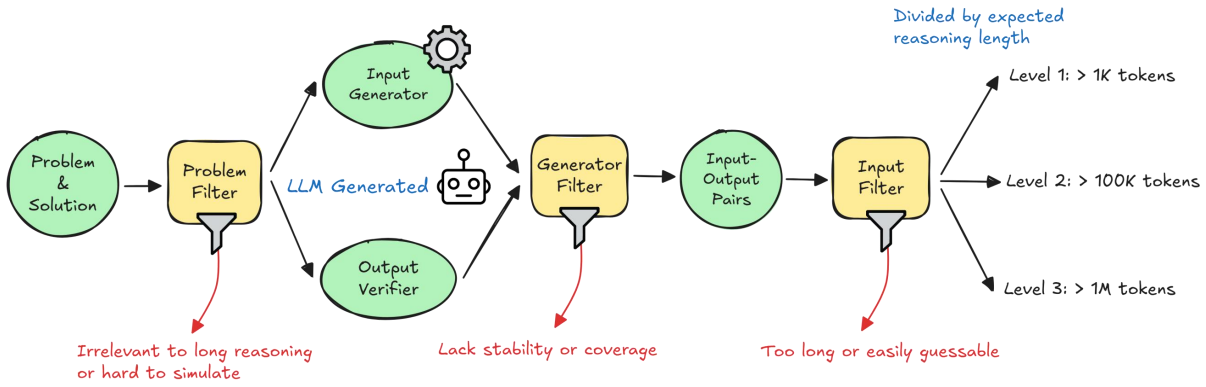
1 Introduction

Recently, Large Language Models (LLMs) have extended the concept of scaling laws from scaling up train-time compute to scaling up test-time compute, achieving remarkable improvements on tasks such as mathematics, programming, and question answering (OpenAI, 2024b; DeepSeek-AI et al., 2025; Team, 2025). To enable test-time scaling, Large Language Models are trained via reinforcement learning to generate extended chain-of-thought that spans thousands of tokens. As a result, their reasoning lengths significantly exceed those of previous approaches (Wei et al., 2022; Yao et al., 2023), introducing new challenges and considerations for evaluation.

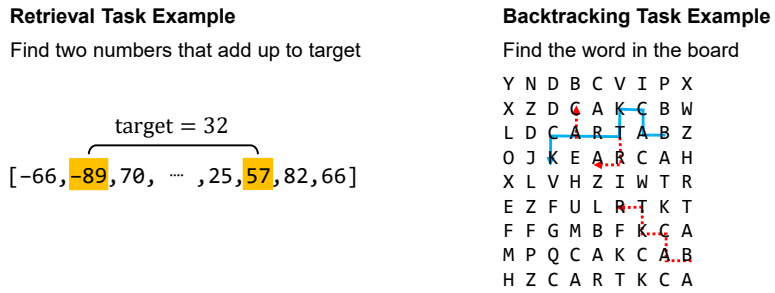
Existing long-context benchmarks focus mainly on long input, such as LongBench (Bai et al., 2024) and Needle-in-a-Haystack (Kamradt, 2023). However, long reasoning differs fundamentally from long input. In the case of long input, the model only needs to passively receive and comprehend information. In contrast, long reasoning requires the model to actively generate, structure, and self-correct its output. Some recent benchmarks have begun to evaluate long-form generation, such as LongGenBench (Liu et al., 2024) and GSM-Infinite (Zhou et al., 2025), but these tasks are relatively simple and remain far from reflecting the complexity of long reasoning processes.

To better evaluate the long reasoning capabilities of the models, we propose LONGREASONARENA, which assesses performance by requiring the models to execute algorithms to solve problems. Executing algorithms inherently requires the model to perform long reasoning, involving a large number of algorithmic steps. The number of required steps and the corresponding reasoning length can be controlled by varying the input, which allows us to arbitrarily scale the required reasoning length. During algorithm execution, some essential capabilities required for long reasoning are put to the test, such as retrieval and backtracking. Research in cognitive science posits that memory and backtracking play important roles to the human reasoning process(Newell et al., 1972; Laird, 2019; Beatty and Vartanian, 2015; Risko and Gilbert, 2016). Empirical observations of LLMs' long reasoning confirm a similar dependency: models must retrieve prior intermediate steps to maintain coherence and employ backtracking to correct errors or switch strategies. These reasoning dynamics are mirrored in the domain of algorithm execution. For instance, algorithmic procedures like depth-first search inherently test a model's ability to perform extensive backtracking. Likewise, the manipulation of data structures—requiring dynamic updates such as insertion and deletion—poses a comparable challenge to the model's ability to manage and retrieve information effectively.

To construct the benchmark, we collect algo-



(a)



(b)

Figure 1: (a) The construction process of LONGREASONARENA (b) Algorithmic problems abstract key elements of long reasoning process, including retrieval, backtracking, and other essential cognitive operations.

rithmic problems and their corresponding solution code from LeetCode. For each problem, we use Qwen2.5-Coder-32B-Instruct (Hui et al., 2024) to generate an input generator function and an output verifier. The input generator is required to stably produce valid inputs and ensure sufficient coverage over the solution code, which enables the generation of an unlimited number of diverse inputs. Problems lacking a qualified generator are excluded from the benchmark. Furthermore, to reduce samples that can be solved through guessing or heuristic methods, we discard any sample where a simple guessing program can get the correct answer.

We categorize tasks into different difficulty levels based on the expected number of reasoning steps. To estimate the reasoning steps required to solve each problem, we measure the number of execution lines when running the solution code on the given input. LONGREASONARENA is divided into three levels: Level 1, 2, and 3 correspond to approximately 1K, 100K, and 1M tokens of reasoning, respectively. LONGREASONARENA poses a significant challenge for both open-source and proprietary LLMs. For example, Deepseek-R1 achieves only 7.5% accuracy on Level 3.

We evaluate seventeen models on LONGREASONARENA, including reasoning and non-reasoning models of various sizes. Based on the analysis of the evaluation results, we reveal several limitations of current reasoning models. First, accuracy decreases as the expected number of reasoning steps increases, following a linear relationship with the logarithm of the step count. Second, models struggle with basic reasoning operations such as retrieval and backtracking: for instance, although models perform well on long-input retrieval tasks, they still often fail at retrieval operations within long reasoning chains, such as checking whether a number appears.

The contributions of this paper are summarized as follows:

- We propose LONGREASONARENA, a benchmark for evaluating long reasoning by algorithmic execution, where the required reasoning length can be arbitrarily scaled (up to 1M tokens) through input control. Our tasks reflect key aspects of long reasoning such as retrieval and backtracking.
- We evaluate seventeen models on LONGREASONARENA. Our analysis reveals that model

accuracy declines linearly with the logarithm of the expected reasoning steps, and that current models still struggle with basic retrieval and backtracking operations.

- We will release LONGREASONARENA to facilitate future research on long reasoning and provide a standardized evaluation framework.

2 Related Work

Long Input Benchmarks Existing long-context benchmarks primarily focus on evaluating LLM's ability to comprehend long inputs, such as the ability to retrieve and summarize (Bai et al., 2024; An et al., 2024; Kamradt, 2023; Hsieh et al., 2024; Zhang et al., 2024). Some papers evaluate models' reasoning ability using a retrieve-then-reason approach (Li et al., 2024; Ling et al., 2025; Kuratov et al., 2024; Zhuang et al., 2025), but the challenge still primarily lies in extracting information from long inputs rather than performing long reasoning. In contrast, LONGREASONARENA explicitly requires models to leverage long reasoning process to solve the task.

Long Generation Benchmarks Recently, several studies have begun to explore the evaluation of models' long generation capabilities. Liu et al.'s LongGenBench (Liu et al., 2024) concatenate multiple questions to construct a single prompt and requires the model to respond to each question within a single response. Wu et al.'s LongGenBench (Wu et al., 2025) tasks the model with sequentially completing a series of subtasks, incorporating specific details at designated positions in the output text. Both benchmarks construct long generation tasks based on relatively simple and repetitive patterns, with subtasks that are independent of one another. GSM-Infinite (Zhou et al., 2025) abstracts GSM-8K problems into computational graphs, which can be manipulated to control over the reasoning complexity and noise level. However, the problem space of GSM-Infinite remains limited to composite arithmetic operations. LR^{2}Bench (Chen et al., 2025) evaluates the long-chain reflective reasoning capabilities of LLMs. However, it only contains six problems, all restricted to constraint satisfaction problems. Previous benchmarks fall short in capturing the complexity of long reasoning processes. In contrast, LONGREASONARENA encompasses a more diverse set of tasks, enabling a broader evaluation across a wider range of reasoning challenges.

Code Execution Some recent works leverage LLM as code executors, tasking the model with simulating line-by-line code execution. Lyu et al.(Lyu et al., 2024) curated a dataset of 200 code snippets along with corresponding input-output examples to evaluate models' abilities in this role. CodeI/O (Li et al., 2025) prompts DeepSeek-V2.5 (DeepSeek-AI et al., 2024) to do inputs/outputs prediction. The generated chain-of-thought is used for supervised fine-tuning other base models. While these works share certain similarities with LONGREASONARENA, they primarily focus on relatively simple tasks and are not designed to evaluate long reasoning capabilities. For example, in CodeI/O, the input and output lists are constrained to a maximum length of 20, and string lengths are limited to 100 characters. Even models lacking long reasoning capabilities, such as DeepSeek-V2.5, can achieve over 50% prediction accuracy. In contrast, LONGREASONARENA is explicitly designed to evaluate long reasoning capabilities, presenting tasks that are challenging even for state-of-the-art reasoning models.

Program Synthesis The program synthesis capability of LLMs has attracted widespread attention. Works such as HumanEval(Chen et al., 2021), MBPP(Austin et al., 2021), and APPS(Hendrycks et al., 2021) have evaluated the program generation ability of LLMs. LiveCodeBench(Jain et al., 2024) further addresses the issue of data contamination. However, these benchmarks are not designed to directly evaluate a model's long reasoning ability, making it difficult to estimate the number of reasoning steps required for solving a problem. For instance, if a model has encountered similar code snippets during training, the reasoning difficulty can be significantly reduced. In contrast, our benchmark requires the model to execute every step correctly in order to arrive at the correct answer, with both controllable and scalable task complexity.

3 Benchmark Construction

3.1 Problem Selection

Our goal is to select diverse problems that reflect key characteristics of long reasoning processes. Our problems are selected from LeetCode , which provides a broad coverage of algorithmic topics and offers detailed information for each problem, including tags, solutions, and input constraints. We first filter the dataset by tags, excluding problems

such as “Database” and “Randomized” that primarily test implementation skills or are hard to simulate with LLMs. To facilitate input generation and output verification, we further exclude problems that have no return value, involve custom classes in type annotations (e.g., "TreeNode"), or involve floating-point numbers in type annotations, which are prone to precision issues.

Backtracking and memory are two fundamental components of long reasoning. Backtracking is a phenomenon unique to long reasoning and does not arise in tasks that only involve processing long inputs. It reflects the model’s ability to explore multiple reasoning paths and reconsider its approach by abandoning the current trajectory and trying alternatives. Memory, on the other hand, is essential for both long reasoning and long input processing. However, in the context of reasoning, memory must be dynamically updated to incorporate new intermediate conclusions and revise previous errors as the thought process unfolds.

To better evaluate the model’s backtracking and memory capabilities, we select a subset of problems as core problems and sample a larger number of inputs for each. For backtracking, we select problems tagged with “Depth-First Search” or “Backtracking.” For memory, we select problems tagged with “Dynamic Programming” or “Breadth-First Search,” as the former requires maintaining and updating a value matrix, while the latter involves managing a dynamically evolving queue. Analysis of LLMs’ reasoning traces across all problems validates that LLMs indeed exhibit more frequent backtracking and memory manipulation behaviors on these core problems (details provided in Section E.2).

3.2 Sample Generation

Input Generator To facilitate the generation of inputs with unbounded quantity and complexity, we leverage Qwen2.5-Coder-32B-Instruct (Hui et al., 2024) to generate an input generator function for each problem. We define two criteria to evaluate the effectiveness of the input generator: stability and coverage. Stability requires that the generator consistently produces valid inputs without triggering errors in the solution function. Coverage requires that the generator produces a diverse set of inputs that maximizes the code coverage of the solution. In practice, we consider a generator to be qualified if it can consecutively generate ten valid inputs and achieve over 90% code coverage. Problems without a qualified generator are excluded from the benchmark. Input generator functions generated by Qwen2.5-Coder-32B-Instruct meet the qualification criteria for 86% of the problems, which is sufficient for our purposes.

Difficulty Level We categorize data samples (problem-input pairs) into different difficulty levels based on the number of execution lines measured when running the solution code on the given input. This metric serves as a proxy for the number of steps required to solve the problem. Specifically, samples with between 10^2 and 10^4 execution lines are labeled as Level 1, those with between 10^4 and 10^5 lines as Level 2, and those with between 10^5 and 10^6 lines as Level 3. Although models may occasionally solve certain samples using fewer steps by guessing or heuristic methods, the average number of reasoning steps at each level remains proportional to the average number of execution lines.

We estimate that each execution line typically requires at least 10 tokens to complete, such as performing a comparison or computation command. Accordingly, solving all Level 2 tasks without guessing would require a model to process at least 100K tokens of reasoning per sample; for Level 3, this requirement increases to at least 1M tokens per sample. These demands far exceed the current capabilities of existing reasoning models. LONGREASONARENA is designed to challenge and guide the development of reasoning models, and to serve as a benchmark for future models capable of handling longer reasoning processes.

Input Length Constraint To ensure that the primary challenge of the task lies in long reasoning rather than long input, we retain only those samples whose input length is within 32K tokens. Dataset statistics are shown in Table 1. Even for Level 3 subset, at least 50% of the samples have input lengths within 1K tokens. Compared to the typical 128K context length supported by current models, this is relatively small, ensuring that the core difficulty of our task does not arise from handling long inputs.

Filtering Out Easily Guessable Samples Some generated samples are categorized as high difficulty due to a large number of execution lines. However, in practice, LLM may arrive at the correct answer using only a small number of reasoning steps by guessing or heuristic methods. This undermines the sample’s ability to effectively evaluate

Table 1: The statistics of datasets at different levels.

	Level 1	Level 2	Level 3
# Problems	262	306	288
# Samples	514	632	523
Median execution line count	1,444	31,592	202,412
90th percentile of execution line count	7,210	80,689	557,175
Median input length	64	524	1,983
90th percentile of input length	570	6,551	22,877

long reasoning capabilities. For example, in a task that asks for all triplets in an array that sum to zero, if such triplets exist, the model must reason through the problem to identify them. But if no such triplets exist, a model may terminate early and guess that the answer is "no solution," simply because it has not encountered any valid triplets during its brief attempts. To eliminate such cases, we prompt Qwen2.5-Coder-32B-Instruct to write a program that guesses the answer using the simplest possible logic. If the guessed answer matches the ground truth, the sample is removed from the dataset. Furthermore, if current reasoning models can stably solve Level 3 samples for a given problem, we consider the problem insufficiently challenging and exclude it from the dataset. Concretely, we generate 5 Level 3 samples per problem and use QwQ's performance as the criterion: if QwQ correctly solves all 5, the problem is discarded.

Output Verification During evaluation, models are prompted to enclose their final answer within `\boxed{}`. Any formatting errors or incorrect answers are treated as incorrect. The reference answer for each sample is obtained by executing the solution code. However, some problems permit multiple correct outputs (e.g., when all combinations or permutations are to be returned in any order). To account for such cases, we use Qwen2.5-Coder-32B-Instruct to determine whether the problem is input-dependent in this regard, and to generate code that verifies whether the model's generated answer is equivalent to the reference answer.

4 Evaluation

4.1 Setup & Results

We evaluate 17 models (OpenAI, 2025, 2024b,a; DeepSeek-AI et al., 2025; Yang et al., 2025; Team, 2025; Anthropic, 2025; Yang et al., 2024; Grattafiori et al., 2024), including 13 open-source models and 4 proprietary models. For all open-source models except DeepSeek-R1 (due to the difficulty of deploying DeepSeek-R1), we perform inference using vLLM (Kwon et al., 2023) on 8 NVIDIA A100 GPUs. For the proprietary models and DeepSeek-R1, we use API calls. Detailed inference settings are provided in Section A.

The evaluation results are shown in Table 2. Overall, large reasoning models significantly outperform non-reasoning models. However, for Level 2 and Level 3 tasks, state-of-the-art reasoning models still struggle. For instance, at Level 3, DeepSeek-R1 attains only 7.5% accuracy, while GPT-5 reaches just 26.4%. We also evaluated human performance on LongReasonArena, finding that it remains challenging for humans as well (see Section C for details).

4.2 Impact of Reasoning and Input Difficulty on Accuracy

We analyze the impact of expected number of reasoning steps and input length on model accuracy, as shown in Figure 2. Accuracy decreases as the expected number of reasoning steps increases, following a strong linear trend with respect to the logarithm of steps. For all reasoning models, $R^2 > 0.9$ (detailed results in Section G). This indicates that our estimation of the required reasoning steps serves as a effective proxy for task difficulty.

With respect to input length, model accuracy varies only slightly when inputs are short, suggesting limited sensitivity to small-scale inputs. However, as the input length increases beyond approximately 10^2 tokens, accuracy drops sharply.

Across both trends, reasoning models (o1, QwQ, Claude 3.7 Sonnet, DeepSeek-R1) exhibit similar patterns of decline, while the non-reasoning model GPT-4o performs consistently worse. This consistency across diverse reasoning models highlights that the observed trends are not model-specific but rather reflect a general property of reasoning mod-

Table 2: Evaluation results of 17 models across difficulty levels.

	Reasoning Model	Level 1	Level 2	Level 3
GPT-5	✓	81.5	41.8	26.4
o1	✓	59.3	29.6	16.4
Qwen3-32B	✓	49.4	19.3	11.3
QwQ	✓	49.4	20.4	10.7
Qwen3-14B	✓	50.4	16.9	8.6
Claude 3.7 Sonnet	✓	44.2	15.5	7.8
DeepSeek-R1	✓	40.1	15.7	7.5
DeepSeek-R1-Distill-Qwen-32B	✓	38.5	13.9	7.5
Qwen3-8B	✓	43.2	16.3	7.1
Qwen3-4B	✓	42.4	13.4	6.7
QwQ-preview	✓	29.0	8.9	3.6
DeepSeek-R1-Distill-Qwen-14B	✓	32.7	9.8	3.3
GPT-4o	×	23.0	5.7	2.1
Qwen2.5-72B	×	20.6	5.2	2.1
DeepSeek-R1-Distill-Qwen-7B	✓	16.3	3.3	1.9
Llama 3.1 70B	×	12.8	3.3	1.2
DeepSeek-R1-Distill-Qwen-1.5B	✓	1.0	0.3	0.0

els.

4.3 Retrieval Failures in Reasoning Process

To analyze the retrieval failures made by reasoning models, we conduct an in-depth analysis using a simple problem, Two Sum. The problem is stated as follows: "Given an array of integers $nums$ and an integer $target$, return indices of the two numbers such that they add up to $target$." The reference solution involves iterating through the array and checking whether $target - nums[i]$ exists in the array. Solving this problem requires only two core capabilities: basic arithmetic (addition and subtraction) and retrieval.

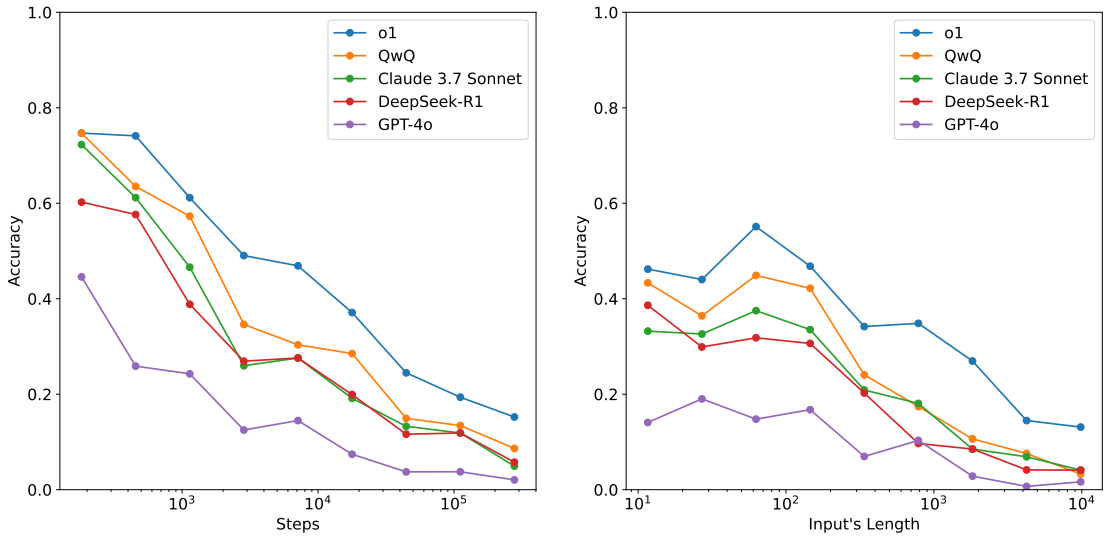
We control the difficulty of the task by varying the length of the input array. Figure 3(a) shows that accuracy decreases as the array length increases, exhibiting a linear relationship with the logarithm of the array length ($R^2 = 0.975$). Given that the expected number of reasoning steps scales linearly with array length, this trend aligns with the pattern observed in Figure 2(a). Notably, the model completely fails the task once the array length reaches 1,000.

To determine whether the primary source of error lies in arithmetic computation or retrieval, we extracted equations from the model's reasoning process and evaluated their correctness. The correctness rate of these equations reaches 98%. Moreover, for this task, the computations at different indices are independent, and the model only needs to perform the correct computation at the index corresponding to the answer in order to succeed. Therefore, there is no accumulation of errors across steps. These findings indicate that the dominant source of failure lies in the model's retrieval ability.

Further examination of erroneous cases reveals that errors can be categorized into index errors and full errors. An index error occurs when the model identifies the correct value pair but retrieves incorrect indices. In the following example, given the input $nums[74] = -5377$, $nums[75] = -22651$, $nums[76] = 27401$, the model correctly computes based on $nums[74]$, but fails to retrieve $nums[75]$. Instead, it mistakenly assigns the value of $nums[76]$ to index 75. If the skipped value is not part of the correct answer, the model may still find the correct value pair; however, this leads to systematic misalignment in all subsequent indexes.

> Index74: -5377 → complement - 22316 → add.
> Index75: 27401 → complement - 55094 → add.

A full error refers to a failure to identify the correct value pair. In the following example, the correct answer is [156, 382] ($target = 1973$, $nums[156] = 31753$, $nums[382] = -29780$).



(a) (b)

Figure 2: Model accuracy with respect to (a) expected number of reasoning steps and (b) input length. Accuracy consistently decreases with both increasing reasoning and input difficulty. A strong log-linear relationship is observed with respect to the expected number of reasoning steps across all reasoning models.

Although the model correctly computes that the number corresponding to -29,780 is 31,753, it fails to retrieve that 31,753 has already appeared in the input.

> Index156:31753 → complement is negative. Not present.
> ...
> Index382:-29780 → complement 1973 + 29780 = 31753. Not present.

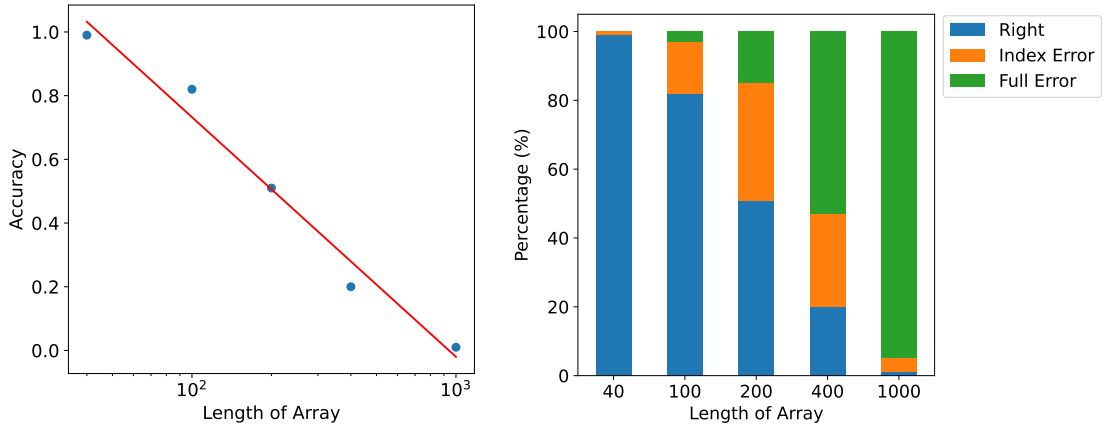
Figure 3(b) shows that when the array length is relatively short, most errors are index errors. However, as the array length increases, the proportion of full errors rises steadily and eventually becomes the dominant error type. This further demonstrates that as the array length grows, the retrieval task becomes increasingly difficult for the model.

Simple retrieval tasks, such as sequentially accessing array elements or determining whether a specific value appears (similar to a Needle-in-a-Haystack task), are typically easy for current models when only a long input is involved. However, when these retrieval operations must be repeatedly carried out throughout a long reasoning process, the models' performance degrade significantly. This highlights an important distinction: although both scenarios involve long contexts, long reasoning processes introduce fundamentally different challenges compared to long inputs. Current models remain constrained by the accuracy of retrieval during reasoning.

4.4 Backtracking Failures in Reasoning Process

To analyze the backtracking failures of reasoning models, we adapt the original Word Search problem to serve as a case study. In our modified version, the task is stated as follows: "Given an m × n grid of characters board and a string word, return the list of positions that form the word in order if the word exists in the grid. If the word does not exist, return an empty list. The word can be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or vertically neighboring. The same letter cell may not be used more than once." In contrast to the original formulation, which simply asks whether the word exists and returns a boolean result, this version enables clearer differentiation between correct reasoning and lucky guesses. Solving this task requires only the ability to perform depth-first search (DFS) and does not involve any arithmetic computation.

We keep the size of the board fixed and vary the length of the word to control the depth of the DFS. As shown in Figure 4, the accuracy declines approximately linearly with increasing word length ($R^2 = 0.961$). When the required search depth reaches 20, the model struggles to complete the task successfully. By analyzing models' reasoning



(a) (b)

Figure 3: The performance trend and error type analysis for the Two Sum task. (a) The accuracy exhibits a linear decline with respect to the logarithm of array length. (b) Error type distribution shows a transition from index errors to full errors as array length increases.

processes, we find that reasoning models exhibit limited backtracking capabilities. They are ineffective at exploring new paths and tend to repeatedly revisit previously explored ones.

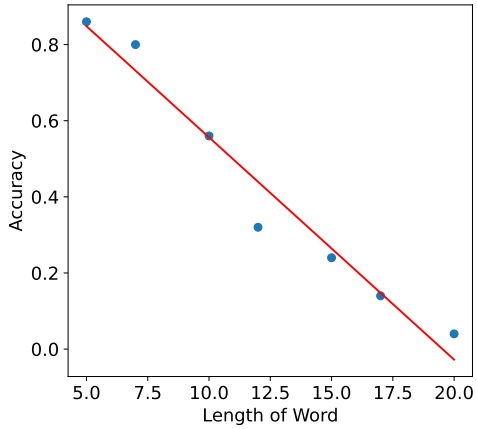


Figure 4: The performance trend for the Word Search task. The accuracy exhibits a linear decline with respect to the length of the word.

To quantify the breadth of exploration, we define a valid path as a cell sequence $S = (x_i, y_i)$ of length greater than two, where each adjacent pair of cells is spatially adjacent and satisfies $board[x_i, y_i] = word[i]$. Furthermore, if a valid path is not a subpath of any other valid path, we classify it as a distinct path. We measure the number of distinct paths discovered during reasoning process and observe that the model discovers only 4.9 distinct paths on average. This small number indicates insufficient exploration. As illustrated in the example in Section F, the model repeatedly checks the sequences (3,10) → (3,9) → (2,9) → (2,8) → (1,8) → (0,8) → (0,7) → (0,6) and (3,10) → (3,9) → (4,9) → (4,8), but fails to explore further. Continuing from (2,8) to (2,7) would have led to the correct path, yet the model fails to backtrack to this point.

5 Conclusion

We present LONGREASONARENA, a benchmark designed to evaluate the long reasoning capabilities of large language models through algorithmic execution. LONGREASONARENA bridges a critical gap left by existing benchmarks focused primarily on long inputs. Our analysis reveals that model performance degrades predictably with increased required steps, and that current models struggle with basic retrieval and backtracking operations. LONGREASONARENA thus provides a scalable and controllable framework for probing and advancing the frontiers of large reasoning models. For future research, we will further broaden the evaluation of models' reasoning capabilities to include more diverse scenarios such as multimodal reasoning.

Limitations

To estimate the number of reasoning steps required for each task, we approximate it using the number of execution lines measured when running the solution code on the given input. While this generally reflects the reasoning complexity of the task, it may deviate from the exact number of steps required by the model, as the model may adopt different strate-

gies to solve the problem. Nevertheless, precisely calculating the number of required reasoning steps for all tasks is infeasible, and our method serves as a sufficiently effective approximation.

Furthermore, evaluation through algorithmic execution primarily focuses on deductive reasoning. It cannot cover all forms of reasoning, such as inductive reasoning or analogical reasoning. Evaluating other types of reasoning remains an important direction for future work.

Broader Impacts

LONGREASONARENA focuses on evaluating models' long reasoning capabilities, which facilitates the extension of their reasoning processes and the enhancement of their reasoning abilities. This leads to greater potential for solving complex real-world problems. However, as reasoning processes become longer and more sophisticated, it also becomes increasingly difficult to interpret and monitor the model's reasoning trajectory, especially in cases involving harmful or deceptive intent.

References

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2024. L-eval: Instituting standardized evaluation for long context language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14388–14411.

Anthropic. 2025. Claude 3.7 sonnet system card.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137.

Erin L Beatty and Oshin Vartanian. 2015. The prospects of working memory training for improving deductive reasoning. *Frontiers in human neuroscience*, 9:56.

Jianghao Chen, Zhenlin Wei, Zhenjiang Ren, Ziyong Li, and Jiajun Zhang. 2025. LR²Bench: Evaluating long-chain reflective reasoning capabilities of large language models via constraint satisfaction problems. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 6006–6032, Vienna, Austria. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, and 138 others. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *Preprint*, arXiv:2405.04434.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and 1 others. 2021. Measuring coding challenge competence with apps. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? In *First Conference on Language Modeling*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*.

Gregory Kamradt. 2023. Needle In A Haystack - pressure testing LLMs. *Github*.

Yury Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. 2024. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Advances in Neural Information Processing Systems*, 37:106519–106554.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

John E Laird. 2019. *The Soar cognitive architecture*. MIT press.

Huayang Li, Pat Verga, Priyanka Sen, Bowen Yang, Vijay Viswanathan, Patrick Lewis, Taro Watanabe, and Yixuan Su. 2024. ALR^2: A retrieve-then-reason framework for long-context question answering. *arXiv preprint arXiv:2410.03227*.

Junlong Li, Daya Guo, Dejian Yang, Runxin Xu, Yu Wu, and Junxian He. 2025. Codei/o: Condensing reasoning patterns via code input-output prediction. *Preprint*, arXiv:2502.07316.

Zhan Ling, Kang Liu, Kai Yan, Yifan Yang, Weijian Lin, Ting-Han Fan, Lingfeng Shen, Zhengyin Du, and Jiecao Chen. 2025. Longreason: A synthetic long-context reasoning benchmark via context expansion. *arXiv preprint arXiv:2501.15089*.

Xiang Liu, Peijie Dong, Xuming Hu, and Xiaowen Chu. 2024. Longgenbench: Long-context generation benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 865–883.

Chenyang Lyu, Lecheng Yan, Rui Xing, Wenxi Li, Younes Samih, Tianbo Ji, and Longyue Wang. 2024. Large language models as code executors: An exploratory study. *arXiv preprint arXiv:2410.06667*.

Allen Newell, Herbert Alexander Simon, and 1 others. 1972. *Human problem solving*, volume 104. Prentice-hall Englewood Cliffs, NJ.

OpenAI. 2024a. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

OpenAI. 2024b. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

OpenAI. 2025. Gpt-5 system card. https://openai.com/index/gpt-5-system-card.

Evan F Risko and Sam J Gilbert. 2016. Cognitive offloading. *Trends in cognitive sciences*, 20(9):676–688.

Qwen Team. 2025. QwQ-32B: Embracing the power of reinforcement learning.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *Preprint*, arXiv:2501.18585.

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

Yuhao Wu, Ming Shan Hee, Zhiqing Hu, and Roy Ka-Wei Lee. 2025. Longgenbench: Benchmarking long-form generation in long context llms. *Preprint*, arXiv:2409.02076.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and 1 others. 2024. ∞ bench: Extending long context evaluation beyond 100k tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277.

Yang Zhou, Hongyi Liu, Zhuoming Chen, Yuandong Tian, and Beidi Chen. 2025. Gsm-infinite: How do your llms behave over infinitely increasing context length and reasoning complexity? *arXiv preprint arXiv:2502.05252*.

Tianyi Zhuang, Chuqiao Kuang, Xiaoguang Li, Yihua Teng, Jihao Wu, Yasheng Wang, and Lifeng Shang. 2025. Docpuzzle: A process-aware benchmark for evaluating realistic long-context reasoning capabilities. *arXiv preprint arXiv:2502.17807*.

A Inference Settings

In this section, we detail the inference configurations used in our experiments. We employ two primary methods for model inference: local deployment via vLLM and remote access via APIs. For open-weights models served by vLLM, we apply

specific generation parameters tailored to reasoning and non-reasoning models, as listed in Table 3. For proprietary models, we adhere to the token limits and constraints outlined in Table 4.

B Evaluation Variance under Different Random Seeds

To assess the impact of random seeds on evaluation results, we re-evaluated QwQ on Level 1 using different seeds. As shown in Table 5, the mean accuracy is 50.9 with a standard deviation of 0.93. The minimum and maximum scores are 53.5 and 49.6, respectively. This indicates that the evaluation results are stable across different random seeds.

Table 5: Evaluation results of QwQ on Level 1 under different random seeds.

Seed	1	2	3	4
Accuracy	51.2	50.2	50.0	51.4
Seed	5	6	7	8
Accuracy	50.2	49.6	50.4	50.2
Seed	9	10	11	12
Accuracy	51.6	51.2	51.4	53.5
Seed	13	14	15	16
Accuracy	50.8	51.4	51.2	50.2

C Human Baseline

We conducted a preliminary human baseline evaluation on a randomly selected subset of our benchmark at Level 1 difficulty. The participants were seven computer science students, who participated on a strictly voluntary basis. Participants acknowledged their understanding of the data usage and agreed to participate by proceeding with the questionnaire. Prior to the task, participants were presented with specific guidelines to ensure standardization. The full text of the instructions provided was: "You are permitted to search for solutions and use computational tools; however, please do not directly execute code to solve the problem as a whole. Please record the time taken to complete the task." Following these instructions, the specific problem statements and their corresponding inputs were displayed to the participants.

In total, 20 questions were evaluated. On average, each question took 4 minutes to complete, and the participants achieved an average accuracy of 75%. This result indicates that our benchmark poses a challenge not only for LLMs but also for human participants with relevant domain knowledge.

D The Use of Large Language Models

In paper writing, we use Large Language Models to polish writing. In experiments, we use Large Language Models both as our research subjects and as tools to generate training data.

E Benchmark Construction Details

E.1 Tag Filtering

We exclude problems tagged with categories such as database, pandas, linked list, multithreading, probability and statistics, randomized, design and interactive. Certain tags like database and linked list primarily test the use of libraries or implementation details, which is not the focus to our evaluation. Problems involving randomized behavior and multithreading are excluded because current LLMs are not well-suited for simulating these behaviors. In addition, since our benchmark is designed for single-turn interactions, we exclude interactive problems that require multi-turn or dynamic input/output exchanges.

E.2 Core Problems

To validate whether models exhibit more desired cognitive behaviors in filtered core problems, we segmented model's reasoning traces and used GPT-4o to assess the presence of backtracking and memory read/write behaviors in each segment. Our analysis revealed statistically significant patterns:

- Problems tagged with DFS/backtracking showed a higher frequency of backtracking behaviors in the traces (40.9% vs. 32.4%, $p < 1e\text{-}4$).
- Problems tagged with BFS/DP showed a higher frequency of memory-related behaviors (87.2% vs. 77.2%, $p < 1e\text{-}4$).

These results suggest that the chosen tags are not arbitrary, but indeed correspond to different reasoning dynamics exhibited by the models.

E.3 Code Coverage

We used Python's coverage library to measure line-level code coverage. Specifically, we tracked the code coverage of the solution code during execution with ten different inputs (excluding import statements as well as class and function definitions).

Table 3: Settings for vLLM Inference

	Reasoning Models	Non-Reasoning Models
MaxNewTokens	32K	8K
System Prompt	N/A	You are a helpful assistant
Temperature	0.6	
TopP	0.95	
MinP	0	
TopK	40	
vLLM Version	0.8.5	

Table 4: Settings for API Calls

	Maximum Number of Reasoning Tokens	Maximum Number of Output Tokens
Deepseek-R1	32K	36K
Claude 3.7 Sonnet	32K	36K
GPT-4o	N/A	default
o1	medium reasoning effort	
GPT-5	medium reasoning effort	

Below is an example of insufficient coverage: due to a bad input generator, all generated samples for this problem exit directly at the first return statement.

```
class Solution:
    def __main(self, grid: List[List[int]]) ->
        int:
        if grid[0][1] > 1 and grid[1][0] > 1:
            return -1

        m, n = len(grid), len(grid[0])
        dist = [[inf] * n for _ in range(m)]
        dist[0][0] = 0
        q = [(0, 0, 0)]
        dirs = (-1, 0, 1, 0, -1)

        ......
```

E.4 Prompt

Prompt for Evaluation

Solve the given problem based on the given inputs.
Don't just reply with code. You should calculate the final answer step by step. Put your final answer within \boxed{}.

Problem:
{problem}

Inputs:
{inputs}

Prompt for Input Generator

Write a Python function called "generate()" to generate test data for a given function. Follow these requirements:
1. Each run of the generate() function should return only one random data sample.
2. The return value of generate() must match the expected input format of the target function.
3. The format and range of the generated test data must follow the specifications in the Constraints section.
4. For any length-related quantities (e.g. list sizes, string lengths, number of elements), values should be sampled uniformly on a logarithmic scale within the allowed range. Notice to avoid undefined behavior with log(0), the lower bound must be at least log(1).

The code block should only include the "generate" function. Think step by step in the comments before the code.

Problem:
{problem}

Constraints:
{constraints}

Examples:
{examples}

Function:
{solution code}

Prompt for Output Verifier

You are a professional Python developer. Complete the following tasks:
Task 1: Determine if the answer should be "order-independent"
I will provide a programming problem statement. Your task is to analyze whether the expected answer should be considered **order-independent**. For example the problem requires returning a **set**, a **list where answer could be in any order**, or a **combination or permutations of elements**.
After analyzing the problem, output either ‘<flag>Order-dependent</flag>"‘ or ‘"<flag>Order-independent</flag>"‘

Task 2: Generate a Boolean Comparison Function
Based on the result of Task 1, generate a Python function ’compare(user_output, expected_output)’, which compares the output of a user’s submission (’user_output’) with the expected answer (’expected_output’) and **returns a boolean value (’True’ or ’False’)**:
- **If the answer is order-independent**, the function should compare the outputs while ignoring order.
- **If the answer is order-dependent**, the function should compare the outputs strictly as given.
The code block should only include the "compare" function.

Problem:
{problem}

Examples:
{examples}

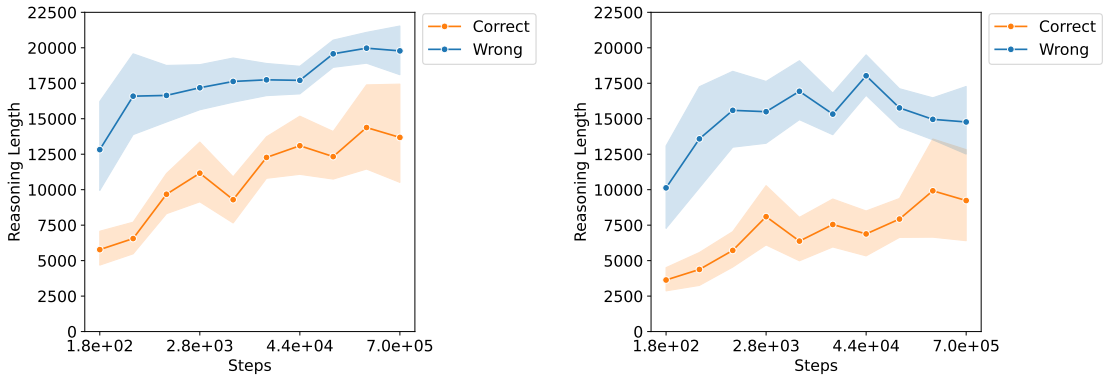
Prompt for Simple Guessing Program

Your task is not to write a standard solution to solve the following problem, but to write the simplest possible function to guess an answer.
If the return value of the problem could mean "no solution exists", prefer to guess that no solution exists. Otherwise, you may use some tricky way to solve the problem, for example hardcoded cases or solutions that work under specific conditions.
The code block should only include the function "guess({params})". Think step by step in the comments before the code.

Problem:
{problem}

F Backtracking Failure Example

So moving left to (4,8). Now, the path is (3,10) → (3,9) → (4,9) → (4,8).
...
So path now is (3,10) → (3,9) → (2,9) → (2,8) → (1,8) → (1,7).
...
Path so far: (3,10) → (3,9) → (2,9) → (1,9).
...
Now the path is (3,10) → (3,9) → (4,9) → (4,8) (J).
...
1. (3,10) 2. (3,9) 3. (2,9) Now, third letter is the second K.
<omit intermediate steps>
Next is U. Left to (0,6) (U)
...
Wait, let me try starting at (3,10) again but take a different route after the two Ks. Path so far: 1. (3,10) T 2. (3,9) K 3. (4,9) K Now, next is J. From (4,9), move left to (4,8) J.
...
(3,10) → (3,9) → (2,9) → (2,8) → (1,8) → (0,8) → (0,7) → (0,6) → then stuck.
...



(a) (b)

Figure 5: Relationship between the expected number of reasoning steps and reasoning length of (a) QwQ and (b) DeepSeek-R1-Distill-Qwen-32B. The reasoning length of correct samples generally increases with higher reasoning complexity, but the reasoning length of incorrect samples is significantly longer than that of correct ones

> Wait let me try one more time with the path starting at (3,10) and see if there's another route. Path: 1. (3,10) T 2. (3,9) K 3. (4,9) K
> ...
> Suppose the path starts at (3,10) → (3,9) K → then to (2,9) K → then to (2,8) J → then to (1,8) H → then to (0,8) Q → (0,7) T → (0,6) U → then need M.
> ...
> 1. (3,10) T 2. (3,9) K 3. (4,9) K 4. (4,8) J

G Summary of Linear Regression Analyses

Table 6: The significance of linear regression between model accuracy and the logarithm of expected number of reasoning steps.

	R^2	p-value
o1	0.981	<1e-4
QwQ	0.946	<1e-4
Claude 3.7 Sonnet	0.914	<1e-4
Deepseek-R1	0.928	<1e-4

H Relationship Between Reasoning Length and Complexity

We analyze the relationship between the model's reasoning length and the reasoning complexity of tasks. As shown in Figure 5, the reasoning length of correct samples generally increases with higher number of required steps. This indicates that the model is, to some extent, able to dynamically adjust its reasoning length according to task complexity. However, it is noteworthy that the reasoning length of incorrect samples is significantly longer than that of correct ones. This suggests that the model could not fully leverage its extended reasoning process—longer reasoning does not guarantee better problem-solving. This observation is also consistent with findings from prior work(Wang et al., 2025).

I License of Artifacts

The LeetCode data is licensed under CC-BY-SA-4.0. Furthermore, all models utilized and evaluated are confirmed to be used in accordance with their respective terms of use. Our benchmark is intended to be used for academic research purposes.