

CHAMP: A Competition-Level Dataset for Fine-Grained Analyses of LLMs' Mathematical Reasoning Capabilities

Yujun Mao
Independent Researcher
amao@bu.edu

Yoon Kim
MIT CSAIL
yoonkim@mit.edu

Yilun Zhou
MIT CSAIL
yilun@csail.mit.edu

Abstract

Recent large language models (LLMs) have shown indications of math deduction abilities. However, it is unclear that for challenging math problems, what information about the problem helps (or hurts). In this paper, we propose a challenging benchmark dataset for such analyses. The Concept and Hint-Annotated Math Problems, or CHAMP, consists of competition-level math problems annotated with “concepts,” or general math facts, and “hints,” or problem-specific tricks. These entities and their interconnections allow us to explore the effects of additional information, such as relevant hints, misleading concepts, or related problems. We conduct 12 preliminary studies with 4 models, summarize our findings and discuss how CHAMP supports general discussions around LLMs’ capabilities to understand and use contexts. The dataset, code and an extended version of the paper are available on the project website at <https://yujunmao1.github.io/CHAMP>.

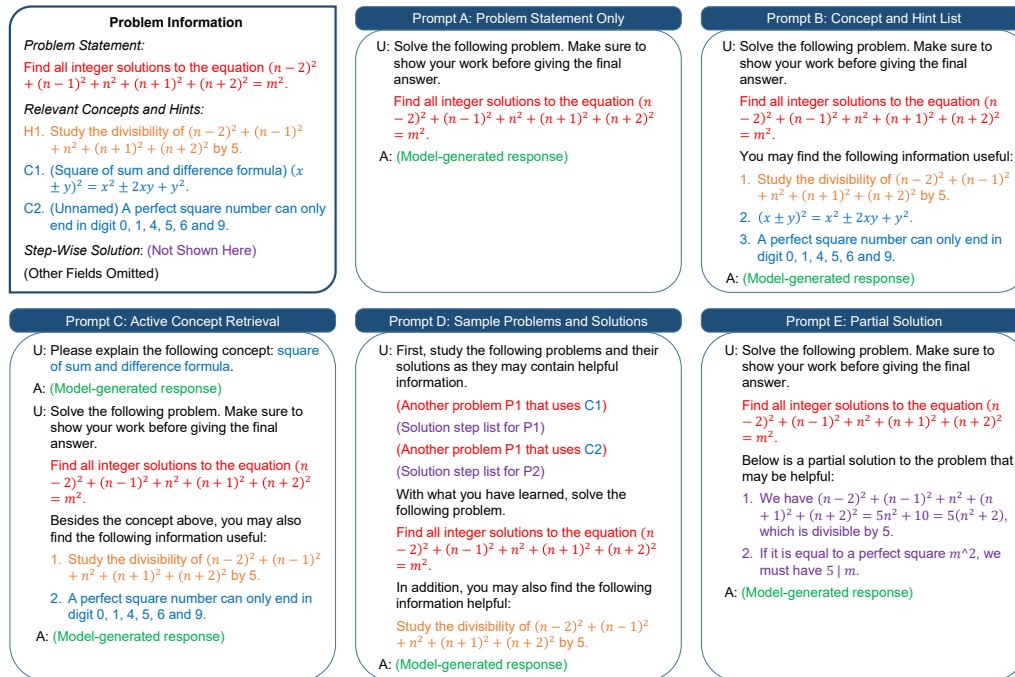


Figure 1: Upper left: an instance in our CHAMP dataset, consisting of a problem statement, relevant concepts and hints, and full step-wise solution. Rest: five prompt templates that we use in the experiment (instantiated into 12 different prompts). U and A refer to the User and Assistant roles.

1 Introduction

Recent large language model (LLMs) have demonstrated superior performance in many tasks that previously required specialized models or were thought to be out of reach of conventional neural networks. One such capability is mathematical reasoning: LLMs can often solve simple math problems and make reasonable attempts at challenging, competition-level problems. In addition to model scaling [6], there are two key factors behind the progress: sophisticated prompting methods such as chain-of-thought [7, 16], tree-of-thought [18] and self-consistency [15], which provide useful heuristics for generating and selecting better reasoning paths, and allowing models to access calculators or code interpreters, which offloads some of the symbolic computation to an external tool [4, 20]. However, one direction remains less explored especially in the context of multi-step problems: how do external pieces of knowledge and hints impact an LLMs’ multi-step reasoning abilities? This is difficult to address with existing datasets which generally consist of problem statements and their solutions, and do not provide annotated concepts or tricks that would be helpful for coming up with the solution.

To enable such analyses, we manually craft the Concept and Hint-Annotated Math Problems (CHAMP), a dataset of 270 problems handpicked from high school competitions and preparation materials, covering five subject categories such as number theory and combinatorics. In addition to problem statements and full solutions, we annotate each problem with two key pieces of information: concepts and hints. The former are general math theorems or formulas, and the latter introduce problem-specific tricks or strategies. The design of CHAMP enables previously under-explored evaluations of multi-step reasoning abilities of LLMs: for example, how helpful are concepts and hints overall? How should we provide such information, a direct presentation or a prompt for the model to recall the content of a concept? Could the model infer useful information from studying sample problems using the same concepts? What happens if the model is provided the wrong concept or hint?

Using this dataset, we design 12 different prompts (Fig. 1), and evaluate OpenAI’s GPT-3.5 Turbo, GPT-4 and GPT-4 Turbo models [13, 14] as well as Google’s Bard model, which is based on PaLM 2 [1]. While we observe a diverse range of behaviors across different models and prompts, the accuracy of the best setting (besides explicitly providing partial solutions) is only 65.2%, showing large potential for improvements with competition-level math for future LLMs and the utility of CHAMP for developing and benchmarking future models.

2 Related Work

Math datasets and benchmarks. Large language models have seen significant improvement in understanding and solving math problems, with GPT-4 [13] being able to tackle most math problems that require grade-level knowledge and direct applications of formulas, even when the problems are presented with a diversity of formats and wording [2]. Nonetheless, SOTA models still struggle with competition-level math problems, such as those found in the MATH dataset [5].

While we also source competition-level problems for which applications of formulas are not straightforward, a key distinction of CHAMP, however, is the information associated with each problem. Most datasets only include the problems and their solutions (given as equations or expressions [10], full text solutions [2, 5] and/or code [12]), but we annotate the concepts and tricks applied to each step of the solution for every problem, as well as the interconnections of problems via shared math concepts. In this way, CHAMP enables fine-grained evaluations of mathematical problem solving abilities of LLMs that are not possible with other datasets. In particular, the construction of CHAMP enables prompting a problem with different additional information, encouraging dataset users to evaluate a model’s behaviors under different contexts.

Reasoning with LLMs. Many techniques have been proposed to improve an LLM’s mathematical reasoning abilities. Much work on this develops prompts to encourage generations of chain-of-thought outputs [16] or selects a final result from multiple (stochastic) outputs [15]. Other approaches prompt LLMs to use external tools such as calculators or Python programs [4] to eliminate arithmetic errors.

Influence of different prompts and contexts. Our dataset and experiments are similar in spirit to works that explore how well LLMs understand different contexts, which have yielded surprising findings. For example, models can be insensitive to label correctness [11] but sensitive to label

distribution [19] and exemplar ordering [8]. McKenzie et al. [9] find that larger LLMs resist absorbing context information inconsistent with world knowledge acquired during training (e.g., redefining $\pi = 432$). Similarly, Wu et al. [17] find that LLMs perform worse in atypical setups for common tasks (e.g., integer addition but in base-9). With the CHAMP dataset, we can explore various behaviors of LLMs and probe into the inner-workings of LLMs, which is still largely unknown.

3 Dataset

We now describe our dataset structure and content. Due to the high level of math expertise required and the need for a unified format, the dataset curation is carried out exclusively by the paper authors.

Problems. Our problems are mainly sourced and adapted from the book *Problem-Solving Strategies* by Engel [3], a classic material for high-school math competitions. We focus on problems that require specific tricks or creative strategies, rather than routine knowledge applications. For ease of performance evaluation, we require a final checkable answer for each problem, and are able to transform some proof problems into this format (e.g., *Prove $f(x) \geq 1$ for $x \geq 0 \rightarrow$ What is the smallest value of $f(x)$ for $x \geq 0$*). Our problems span five categories: number theory, polynomial, sequence, inequality, and combinatorics. App. C contains category-specific notes and considerations during our curation.

For each problem, we manually verify and write the full step-wise solution in natural language, as the solution manual often skip steps, and occasionally contain typographical errors. A final answer (the short version of the full solution such as a short expression or value, or a simple yes/no depending on the answer format) is also explicitly recorded.

Concepts and hints. The key component of our dataset is the annotation of problems with relevant *concepts* and *hints*. Both of which provide helpful information to solve a problem. The distinction between concepts and hints is that the former is about general math knowledge, such as an equation or a theorem, for example, “ $x^2 - y^2 = (x + y)(x - y)$ ”, while the latter are problem-specific tricks or strategies, such as “Add $2xy$ to $x^2 + y^2$ ” (to make a perfect square). For each problem, the full solution is augmented with step-wise annotations of applied concepts and hints, with a full list of concepts and hints reflecting the order of their application. Notice that duplicates are allowed on the list since such information can be re-used in different parts of a solution.

Each concept is additionally annotated with three metadata fields; the *category*, such as “number theory” or “polynomial”, the *name*, such as “difference of squares formula” for the concept $x^2 - y^2 = (x + y)(x - y)$; and the *parent concept*, which is a general form of the concept. For example, both $(x + y)^2 = x^2 + 2xy + y^2$ and $(x + y)^3 = x^3 + 3x^2y + 3xy^2 + y^3$ have the parent concept of the binomial theorem, which states $(x + y)^n = \sum_{k=0}^n \binom{n}{k} x^{(n-k)} y^k$. While every concept is assigned a category, not all concepts have a name or a parent concept.

A main feature of our dataset problems is the lack of a clear line of attack from the beginning. Instead, there are many plausible approaches, leading to different subsequent choices and resulting in a tree of paths to take, with only one (or few) successful ones in the end. Thus, concepts and hints can serve as “branch indicators” that when understood correctly, should guide the model onto the successful path. For the current LLMs, which are largely based on next-token generation algorithms, good comprehension of messages delivered by branch indicators should increase model performance.

Dataset statistics. Overall, we collect 270 problems, 54 concepts and 330 hints across five categories – number theory: 80, polynomial: 50, sequence: 50, inequalities: 50, combinatorics: 40. Since the problems are non-routine, competition-level problems, all the problems require at least 1 concept or hint to solve, with an average of 1.4 concepts and 1.7 hints applied per problem. In addition, each problem has on average 6.0 solution steps, further demonstrating the difficult nature of our dataset. As a MWP dataset, each problem statement has an average of 20.2 words, with an average solution step of 10.9 words. Fig. 2 of App. A plots the distribution of these statistics.

4 Experiment

4.1 Setup

We evaluate GPT-3.5 Turbo (16k context version), GPT-4, GPT-4 Turbo and Bard. For GPT models, we set the temperature to 0 and the maximum number of tokens to generate to 3,000, which is more than enough to output the correct reasoning, and treat any unfinished generation as incorrect. Following mainstream practices, we ask the model to show the work first, before generating the final answer. However, to facilitate answer grading, we additionally ask the model to summarize its answer in a new round of conversation with “Now, summarize the answer above in one sentence, without any intermediate steps or explanations.”

Answer grading. In our experiment, we focus only on the correctness of the final answer (e.g., “yes”, “ $x = 1$ ”, “no solution”, etc.) while leaving the analyses of intermediate derivations to future work. Nonetheless, the same answer can be expressed in multiple ways, such as “no solution”, “none”, “unsolvable”, “impossible to solve”, etc., making simple criteria-based exact or sub-string match to a manually constructed set of valid answers [2, 5] prone to false negatives. Thus, we propose to use GPT-4 as an automated grader, with the prompt shown in Tab. 2 in App. B. To validate this approach, 700 answers were graded by both humans (paper authors) and GPT-4. Around 2% answers receive different judgments, evenly split between human errors and GPT-4 errors. This result establish a 99% accuracy of GPT-4 grading, on par with human accuracy, justifying its use in experiments.

Majority guess baseline. Sometimes models can output correct final answers with incorrect reasoning, especially when those final answers appear a lot in the pre-training corpus. To contextualize a model’s final answer accuracy, we construct the majority guess baseline as follows. For each of the four answer types in our dataset – numeric answers (e.g., 0), expression answers (e.g., n), yes/no answers (e.g., no) and enumeration answers (e.g., *none*) – the baseline guesses the answer values that appear most frequently in the dataset for each answer type. Thus, LLMs exhibiting good math deduction abilities should perform above this baseline. The baseline accuracy is calculated to be 33.0% with details shown in Tab. 5 of App. D.

4.2 Prompts

To demonstrate the wide range of applications of our dataset, we develop 12 different prompts to evaluate models. The first prompt provides only the problem statement, which serves as a model accuracy baseline. The other 11 prompts, on the other hand, add various types of additional information and/or instructions to use such information, as listed below:

1. Providing no additional information beside the problem statement (Fig. 1, Prompt A).
- 2-7. Providing relevant concepts and/or hints in six ways: ordered relevant concepts and hints, unordered relevant concepts and hints, concepts only, hints only, relevant concepts and hints but with the former replaced by their most general version (up the parent-child link), and relevant hints but misleading (i.e., other) concepts from the same category (Fig. 1, Prompt B).
- 8-9. Asking the model to retrieve concept-related information in two ways: 1. recall the content of a concept by its name (while providing the content of unnamed ones directly); 2. give an example to apply the concept. In both cases, another round of conversation provides a list of useful hints. (Fig. 1, Prompt C).
10. Providing sample problems using the same concepts along with their solutions, rather than providing the concepts directly (Fig. 1, Prompt D).
- 11-12. Providing 33.3% and 66.7% ground truth solution steps (Fig. 1, Prompt E).

For all prompting, the same system prompt of “You are an expert on mathematics.” is used. For the Bard model, which does not support system prompt customization, we prepend the system prompt to the beginning of the first user prompt.

4.3 Results

Model performance on various prompts are summarized in Tab. 1. Due to limited API access, only three prompts are tested on Bard. We have several findings. First, GPT-4 and GPT-4 Turbo outperform the majority guess baseline by a large margin, while GPT-3.5 Turbo and Bard struggle. The evolution of performance among GPT models is also evident. Nonetheless, the best performance (besides explicitly providing partial solutions) is 65.2%, which is achieved by GPT-4 Turbo with an ordered

Model	Problem Only	Concept and Hint (C+H) List						Concept Retrieval		Sample Problem	Partial Solution	
		Ordered	Random	C Only	H Only	Root C	Mislead C	Name	Example		33.3%	66.7%
GPT-3.5 T	28.9	33.3	34.4	28.9	33.3	34.1	31.5	32.2	31.1	35.6	33.0	40.4
GPT-4	42.2	53.0	50.0	43.0	51.5	49.6	48.9	50.0	51.5	50.4	54.1	65.6
GPT-4 T	58.1	65.2	60.0	57.8	61.9	63.7	56.3	62.6	63.3	64.1	60.7	64.8
Bard	28.5	33.3	-	-	30.0	-	-	-	-	-	-	-

Majority Guess Baseline: 33.0

Table 1: Final answer accuracy (in percentage) with different prompt settings. The majority guess baseline and all prompts are discussed in Sec. 4.1. “T” in model name refers to Turbo. Bard is only evaluated on a few settings due to limited access.

list of concepts and hints, showing ample room for improvements, demonstrating CHAMP’s potential in benchmarking future models.

In addition, in all cases, the effects of providing the concepts alone (C Only) is minimal, most likely because they are general math knowledge that have been encountered many times during models’ pre-training. Indeed, as Fig. 3 of App. E shows, the model can automatically recall and use relevant concepts even without being instructed. By comparison, providing hints are more helpful since they are created during dataset construction and less likely to be encountered during pre-training, thus increasing the difficulty of problems when not given. It is worth noting that while the models are able to comprehend hints to get to the correct solution path, they are useful only to an extent, with examples showing wrong deductions following correct applications of hints in Fig. 4 of App. E.

Compared to its predecessors, GPT-4 Turbo is notably aligned with human expectation in that it suffers the most from misleading prompts. This shows that the model is good at absorbing new information, even wrong ones, to the point that it makes incorrect reasoning to justify its usage, as shown in Fig. 5 of App. E. Furthermore, we also notice an intriguing behavior of self-correction with GPT-4 Turbo, where the model can identify its mistakes along the way and re-route to a correct path, as shown in Fig. 6 of App. E. Even though GPT-4 Turbo shows the best overall performance, revealing partial solutions helps it the least: while providing partial solutions helps increase all model performance, there is only a 11.5% relative increase from seeing no solution at all to seeing 66.7% of the solutions for GPT-4 Turbo. At the same time, GPT-3.5 Turbo and GPT-4 achieve relative increases of 39.8% and 55.5%, respectively.

Last, we acknowledge limitations in our use of final answer accuracy as the metric for model evaluation. As Fig. 7 of App. E shows, the model could use incorrect reasoning to achieve the correct final answer, making final answer accuracy a somewhat inflated measurement of the model’s true reasoning ability. While this is the predominant practice for most benchmarks [e.g. 2, 5], we leave further exploration and assessment of this issue to future work.

5 Discussion

In this paper, we present a novel dataset, Concept and Hint-Annotated Math Problems (CHAMP), consisting of challenging high school competition-level problems on which current large language models struggle. For each problem, we annotate the necessary concepts, tricks and strategies. The interconnections among those entities allow fine-grained analyses of LLMs’ performance under different context.

The application of the dataset is demonstrated by experiments with 12 prompt designs on four different models. We find that both GPT-4 and GPT-4 Turbo exhibit above-random math problem solving behaviors, with a good level of capability to incorporate useful information, especially hints, to improve model performance. The GPT series of models show clear evolution of performance, with GPT-4 Turbo most aligned with human expectations.

Overall, the unique construction and detailed annotations of CHAMP allow users to explore new prompts beyond the 12 examples presented in this paper. For example, users can develop new prompt designs with information of the step-wise annotations of concepts and hints in the solutions. Besides, future efforts could include model fine-tuning with CHAMP, as well as more systematic qualitative analyses on model outputs. With the latest model of GPT-4 Turbo showing a large room for improvements, we expect CHAMP to be a valuable resource for the AI community to develop and benchmark new models in the future.

References

- [1] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.
- [2] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [3] Arthur Engel. *Problem-solving strategies*. Springer Science & Business Media, 2008.
- [4] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.
- [5] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.
- [6] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- [7] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.
- [8] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*, 2021.
- [9] Ian R McKenzie, Alexander Lyzhov, Michael Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Aaron Kirtland, Alexis Ross, Alisa Liu, et al. Inverse scaling: When bigger isn’t better. *arXiv preprint arXiv:2306.09479*, 2023.
- [10] Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. *arXiv preprint arXiv:2106.15772*, 2021.
- [11] Sewon Min, Xinxu Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.
- [12] Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, et al. Lila: A unified benchmark for mathematical reasoning. *arXiv preprint arXiv:2210.17517*, 2022.
- [13] OpenAI. Gpt-4 technical report. *ArXiv*, 2023.
- [14] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.
- [15] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
- [16] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

- [17] Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. *arXiv preprint arXiv:2307.02477*, 2023.
- [18] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.
- [19] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR, 2021.
- [20] Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, et al. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification. *arXiv preprint arXiv:2308.07921*, 2023.

A Dataset Statistics Visualization

Fig. 2 shows the distributions of various key statistics of the dataset.

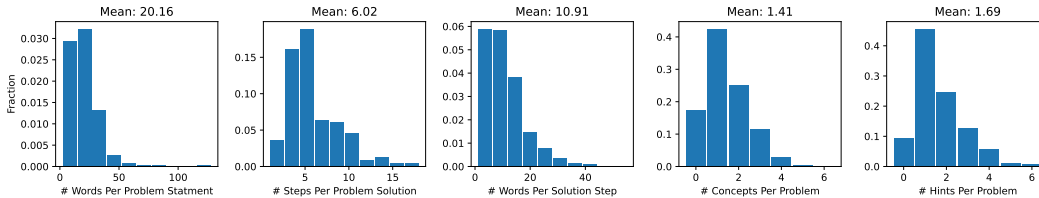


Figure 2: Distributions of key statistics of the dataset.

B GPT-4 Grader Prompt

Tab. 2 shows the grading prompt that we use. We find that GPT-4 is excellent at following the instruction and only outputting one of “correct,” “incorrect” and “partially correct,” with the last judgment common for answers with missed solution or incomplete case discussion. For the purpose of accuracy evaluation, we treat “partially correct” as “incorrect.”

Role	Message
System	You are a math teacher and need to grade student’s homework. For each question, you need to determine the correctness of the answer, given the reference answer. You only need to judge the correctness of the final answer, and should not consider any reasoning or explanations given in the answer. Note that if the student gives an obviously equivalent solution to the reference answer (e.g., 1.5 vs 3/2 or $a^2 - b^2$ vs $(a + b)(a - b)$), the answer should be judged as correct. Your decision should be one of “Correct”, “Incorrect” or “Partially correct”. There is no need to explain your decision.
User	The question is: (Problem statement) The reference answer is: (Ground truth final answer) The student answer is: (Model-generated and summarized final answer) Is the student answer correct, incorrect, or partially correct?
Assistant	(Grading judgment output)

Table 2: The prompt setup used for automated grading by GPT-4.

C Problem Collection and Annotation Considerations

C.1 Number Theory

A notable feature of number theory problems is that most of them are given in the form of proof problems. We manage to convert most of them into problems asking for an answer, with examples listed in Tab. 3. In addition, there are some questions which require non-trivial factorization. Since GPT is often bad at arithmetics above 100, we provide them directly as hints, such as $1971 = 27 \times 73$.

Before	After
Prove that $n^4 + 4^n$ can never be a prime number for integer $n > 1$.	For how many integers n in $\{1, 2, \dots, 99\}$ is $n^4 + 4^n$ a prime number? (Answer: 1)
Prove that $x^2 + y^2 + z^2 = 2xyz$ has no positive integer solutions.	Find all positive integer solutions to the equation $x^2 + y^2 + z^2 = 2xyz$. (Answer: None)
Prove that $323 \mid 20^n + 16^n - 3^n - 1$ for even n .	What are possible values of $20^n + 16^n - 3^n - 1 \pmod{323}$ for even n ? (Answer: 0)

Table 3: Conversion of number theory proof problems into those requiring answers.

C.2 Polynomial

Some polynomial problems require factorization or root finding involves nontrivial arithmetics, similar to number theory problems. To reduce errors in this process, we provide the relevant arithmetic calculation as hints, such as $264 = 6 \times 44$ when factoring $v^2 - 50v + 264 = (v - 6)(v - 44)$.

In addition, there are several polynomial division and remainder problems, for which we provide the concrete definition as a concept (although all models could easily retrieve and explain this definition with a straightforward query of “What is polynomial division and remainder?”):

When a polynomial $f(x)$ is divided by a polynomial $g(x)$, the quotient $q(x)$ and the remainder $r(x)$ are polynomials such that $f(x) = g(x)q(x) + r(x)$ and the remainder $r(x)$ has degree less than that of $g(x)$.

C.3 Sequence

A common type of problems in sequence is to find its limit. However, a prerequisite is to prove that the limit exists. Thus, we frame such questions explicitly, using wording such as “Determine if the limit exists, and if so, find its value.” We also annotate these questions with concept stating the theorem that establish the existence of the limit, most commonly the monotone convergence theorem:

A sequence that is monotonic and bounded has a limit. Specifically, a sequence that is monotonically increasing and bounded from above, or monotonically decreasing and bounded from below, has a limit.

In addition, a common strategy is induction, which shows that a property holds for all a_n by showing that it holds for a_n if it holds for all of a_1, \dots, a_{n-1} . Because the instantiation of the strategy, especially the property to show, is problem-specific, we provide it as a hint, rather than a concept.

C.4 Inequality

Just like with the category of number theory problems, many problems in inequality are written as proofs of inequality identity. We manage to convert them into questions requiring numerical answers with approaches such as asking for the extremum (i.e., maximum or minimum depending on the original inequality) value (while making sure that the value can indeed be attained by some variable value assignment). Some sample conversions are listed in Tab. 4.

C.5 Combinatorics

Most combinatorics problems are verbal, since they often describe a real-world scenario. When applicable, we try to provide the problem with the unmentioned commonsense knowledge (e.g., “On a chess board, two rooks are placed peacefully if they are not on the same row or column.”) before

Before	After
Prove that, for $a, b, c > 0$, $\sqrt[3]{abc} \leq \sqrt{(ab + bc + ca)}/3$?	For positive a, b, c , what is the smallest value of $\sqrt{ab + bc + ca}/\sqrt[3]{abc}$? (Answer: $\sqrt{3}$)
If $n > 1$, prove that $1/(n + 1) + 1/(n + 2) \dots + 1/(2n) > 1/2$.	For how many values of n in $\{101, \dots, 1000\}$ is $1/(n + 1) + 1/(n + 2) + \dots + 1/(2n) > 1/2$? (Answer: 900)
The product of three positive reals is 1. Their sum is greater than the sum of their reciprocals. Prove that exactly one of these numbers is > 1 .	The product of three positive real numbers is 1, and their sum is greater than the sum of their reciprocals. How many of them can be greater than 1? (Answer: 1)

Table 4: Conversion of inequality proof problems into those requiring answers.

the actual problem (e.g., “For an $n \times n$ chess board, find the number of ways that n rooks can be placed peacefully (i.e., any two are placed peacefully).”).

In addition, many combinatorics problems ask for the number of ways in a setup of size n (e.g., the number of ways that n horses can finish in a race with the possibility of ties), and it is solved in the following manner:

1. Find a recurrence relationship to express $P(n)$ in terms of $P(n - 1)$ and $P(n - 2)$ (and possibly more terms), where $P(n)$ is the quantity asked in the question.
2. Find the initial values $P(1), P(2)$ (and possibly more terms).
3. Set up a characteristic equation (which is a polynomial) and find its root.
4. Use the roots to express $P(n)$ as a function of n .

The key difficulty is the root-finding part, so instead of asking for the general expression of $P(n)$ in terms of n , we ask for a specific value, such as $P(7)$, which could be worked out instead by repeatedly applying the recurrence relationship from the initial values. We also make sure that the asked $P(n)$ value is relatively small, usually less than 200, to minimize the chance of arithmetic errors.

D Majority Guess Baseline Construction

Tab. 5 presents a breakdown of problems by the types of answer format and the corresponding baseline answer based on the majority ground truth answer of each answer format.

Answer Format	# Problems	Example	Baseline Answer
Numeric	162 (60.0%)	Let n be 22...22, with a total of 1980 digits. What is $n \bmod 1982$?	0
Expression	45 (16.7%)	In how many ways can you take an odd number of objects from n (distinct) objects?	Sum of all variables (e.g., $n + k$)
Yes/No	42 (15.6%)	Is $4^{545} + 545^4$ a prime number?	No
Enumeration	21 (7.8%)	Find all integer solutions to the equation $x^3 + x^2y + xy^2 + y^3 = 8(x^2 + xy + y^2 + 1)$.	None

Table 5: The construction of baseline answers based on four answer formats.

E Illustrative Examples of Various Reasoning Characteristics

Fig. 3-7 shows various model behaviors, with explanations given in the respective caption and relevant pieces color-highlighted.

Problem: For positive numbers a, b, c , what is the smallest value of $(a + b)(b + c)(a + c)/(abc)$?

GPT-4 Solution (Problem Only Prompt):

To solve this problem, we can use the **AM-GM inequality (Arithmetic Mean-Geometric Mean Inequality)**, which states that the arithmetic mean of a set of non-negative numbers is always greater than or equal to the geometric mean of the same set.

Let's apply this inequality to the given problem:

We have three terms: $a + b$, $b + c$, and $a + c$.

The arithmetic mean of these terms is:

$$((a + b) + (b + c) + (a + c))/3 = (2a + 2b + 2c)/3 = 2(a + b + c)/3$$

(omitted text)

So, the smallest value of $(a + b)(b + c)(a + c)/(abc)$ is achieved when $a = b = c$, and it is equal to 8.

Figure 3: An instance where the model correctly applies a concept spontaneously (green), without being told.

Problem: A function $f(x)$ satisfies $f(x + 1) + f(x - 1) = \sqrt{2}f(x)$. Determine whether the function is periodic, and if so, find its period.

Concept and Hint:

1. (Hint) Let $f(x - 1) = a$ and $f(x) = b$, and study expressions for $f(x + 2)$, $f(x + 3)$,

GPT-3.5 Solution (Ordered Concept and Hint List Prompt):

To determine whether the function is periodic, we need to find a pattern in the values of $f(x)$ as x increases.

Let's start by substituting $f(x - 1) = a$ and $f(x) = b$ into the given equation:

$$f(x + 1) + f(x - 1) = \sqrt{2}f(x)$$

Substituting the values, we get:

$$f(x + 1) + a = \sqrt{2}b$$

Now, let's substitute $f(x + 1) = c$ and $f(x) = b$:

$$c + a = \sqrt{2}b$$

Rearranging the equation, we have:

$$c = \sqrt{2}b - a$$

Now, let's substitute $f(x + 2) = d$ and $f(x + 1) = c$:

$$d + c = \sqrt{2}c$$

(omitted text and wrong final answer)

Figure 4: An instance where the model knows to follow the provided hint (green) but proceeds with an incorrect derivation (red).

Problem: What is the smallest value of $a/(b+c) + b/(a+c) + c/(a+b)$ for positive numbers a, b, c ?

Concepts and Hints:

1. (Relevant Hint) Add 1 to each term in the sum to make the numerator the same and factorize it.
2. (Misleading Concept) For positive x , $x + 1/x \geq 2$, with equality if and only if $x = 1$.

GPT-4-Turbo Solution (Misleading Concept Prompt):

Let's follow the hints provided and solve the problem step by step.

We are given the expression:

$$\frac{a}{b+c} + \frac{b}{a+c} + \frac{c}{a+b}$$

According to the first hint, we add 1 to each term:

$$\left(\frac{a}{b+c} + 1\right) + \left(\frac{b}{a+c} + 1\right) + \left(\frac{c}{a+b} + 1\right)$$

Now, we can rewrite each term to have the same numerator as the corresponding denominator:

$$\frac{a+b+c}{b+c} + \frac{a+b+c}{a+c} + \frac{a+b+c}{a+b}$$

We can factor out the common term $a + b + c$:

$$(a+b+c) \left(\frac{1}{b+c} + \frac{1}{a+c} + \frac{1}{a+b} \right)$$

Now, let's use the second hint. For positive x , we know that $x + 1/x \geq 2$, with equality if and only if $x = 1$. We can apply this inequality to each term in the parentheses:

$$\frac{1}{b+c} \geq \frac{2}{a+b+c}, \frac{1}{a+c} \geq \frac{2}{a+b+c}, \frac{1}{a+b} \geq \frac{2}{a+b+c}$$

Adding these inequalities together, we get:

(omitted text and wrong final answer)

Figure 5: An instance where the model falls victim to a misleading concept (orange) and proceeds with an incorrect derivation (red).

Problem: Find all possible positive integer values of x such that x , $x + 10$ and $x + 14$ are all prime numbers.

GPT-4-Turbo Solution (Concept Application prompt):

To solve this problem, we need to find all positive integer values of x such that x , $x + 10$, and $x + 14$ are all prime numbers. We will use the concept of modular arithmetic to help us narrow down the possibilities.

First, let's consider the three expressions modulo 3. We know that any integer n is congruent to 0, 1, or 2 modulo 3. That is, $n \bmod 3$ is either 0, 1, or 2.

If $x \bmod 3 = 0$, then x is divisible by 3 and cannot be prime unless $x = 3$. However, if $x = 3$, then $x + 10 = 13$ (which is prime) but $x + 14 = 17$ is not prime, so $x = 3$ is not a solution.

(omitted text)

However, we made a mistake in our initial analysis. Let's correct it:

If $x = 3$, then $x + 10 = 13$ and $x + 14 = 17$, which are both prime. So $x = 3$ is indeed a solution.

(omitted text)

Therefore, the only solution is $x = 3$, where all three numbers x , $x + 10$, and $x + 14$ are prime. There are no other positive integer values of x that satisfy the given conditions.

Figure 6: An instance where the model makes a mistake (red) but corrects it later (green).

Problem: For how many integers n in $\{1, 2, \dots, 99\}$ is $n^4 + 4^n$ a prime number?

GPT-4 Solution (Problem Only Prompt):

To solve this problem, we need to check each integer n in the set $\{1, 2, \dots, 99\}$ to see if $n^4 + 4^n$ is a prime number. A prime number is a number that has exactly two distinct positive divisors: 1 and itself.

Let's start with the first few numbers:

1. For $n = 1$, $n^4 + 4^n = 1^4 + 4^1 = 1 + 4 = 5$, which is a prime number.
2. For $n = 2$, $n^4 + 4^n = 2^4 + 4^2 = 16 + 16 = 32$, which is not a prime number.
3. For $n = 3$, $n^4 + 4^n = 3^4 + 4^3 = 81 + 64 = 145$, which is not a prime number.
4. For $n = 4$, $n^4 + 4^n = 4^4 + 4^4 = 256 + 256 = 512$, which is not a prime number.

We can see a pattern here. For $n > 1$, $n^4 + 4^n$ is even (since both n^4 and 4^n are even), and the only even prime number is 2. Therefore, for $n > 1$, $n^4 + 4^n$ cannot be a prime number.

So, the only integer n in the set $\{1, 2, \dots, 99\}$ for which $n^4 + 4^n$ is a prime number is $n = 1$. Therefore, there is only 1 such integer.

Figure 7: An instance of a correct answer from an incorrect reasoning (red).