# TreeCut: A Synthetic Unanswerable Math Word Problem Dataset for LLM Hallucination Evaluation

**Jialin Ouyang**
Columbia University
jo2559@columbia.edu

## Abstract

Large language models (LLMs) now achieve near-human performance on standard math word problem benchmarks (e.g., GSM8K), yet their true reasoning ability remains disputed. A key concern is that models often produce confident, yet unfounded, answers to unanswerable problems. We introduce TreeCut, a synthetic dataset that systematically generates *infinite* unanswerable math word problems and their answerable counterparts, by representing each question as a tree and removing chosen necessary conditions. Experiments show TreeCut effectively induce hallucinations in large language models, including GPT-4o and o3-mini, with rates of 64% and 44% in their respective worst-case scenarios under zero-shot setting. Further analysis highlights that deeper or more complex trees, composite item names, and removing necessary condition near the middle of a path all increase the likelihood of hallucinations, underscoring the persistent challenges LLMs face in identifying unanswerable math problems. The dataset generation code and sample data are available at https://github.com/j-bagel/treecut-math.

## 1 Introduction

Mathematical reasoning is a crucial part of human intelligence. Recent years have witnessed remarkable advancements in the mathematical reasoning capabilities of large language models (LLMs). By leveraging techniques such as chain-of-thought prompting (Wei et al., 2022), state-of-the-art LLMs (e.g., Achiam et al. (2023); Team et al. (2024); Dubey et al. (2024)) achieved human-level performance on benchmarks like GSM8K (Cobbe et al., 2021). However, it remains controversial whether this performance implies reasoning capability beyond pattern matching.

A substantial body of research highlights the capability of Large Language Models in mathematical reasoning. Achiam et al. (2023); Team et al. (2024); Dubey et al. (2024); Yang et al. (2024), among others, achieved over 90% accuracy on GSM8K (Cobbe et al., 2021), a dataset consists of 8K grade school math word problems. Yang et al. (2024); Zhou et al. (2023), among others, achieved over 80% accuracy on the more difficult MATH dataset (Hendrycks et al., 2021), which consists of 12.5K high school math competition problems.

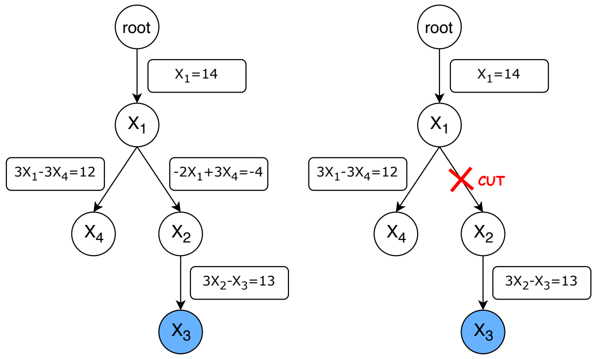
Meanwhile, there is a line of research questioning the reasoning ability of LLMs by showing their vulnerability under superficial changes of the input that do not alter the underlying logic. Works like Shi et al. (2023); Jiang et al. (2024) find that LLMs are easily distracted by irrelevant context or token level perturbation that does not change the underlying logic of the reasoning task. Mirzadeh et al. (2024) further demonstrate that the performance of LLMs declines when numerical values are altered in the questions from the GSM8K dataset.

There is yet another line of research that challenges the ability of LLMs to refrain from answering unanswerable problems. Ma et al. (2024); Li et al. (2024); Sun et al. (2024); Zhou et al. (2024a); Saadat et al. (2024) introduce minor modifications to existing math word problems to create unanswerable variants, and find that LLMs often generate hallucinatory answers for these unanswerable questions, even when they perform well on the original answerable datasets. However, these efforts rely on pre-existing math word problem sources, making them susceptible to training data contamination, limited in scope, and lacking rich structures for extended research.

To address these shortcomings, we propose TreeCut, a synthetic dataset capable of systematically generating an infinite number of unanswerable math word problems and their answerable counterparts. TreeCut considers problem represented by a tree, with nodes representing variables and edges representing formulas. Unanswerable problems are generated by removing an edge along



*Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1073–1085




**Question:**
A burger costs 14 dollars. ~~3 scrambled eggs cost 4 dollars less than 2 burgers.~~ 3 pies cost 12 dollars less than 3 burgers. A BLT sandwich costs 13 dollars less than 3 scrambled eggs. Question: how much does a BLT sandwich cost?

**Solution to the answerable problem:**
It is given as a fact that a burger costs 14 dollars. Combine with the fact that 3 scrambled eggs cost 4 dollars less than 2 burgers, we get a scrambled egg costs 8 dollars. Combine with the fact that a BLT sandwich costs 13 dollars less than 3 scrambled eggs, we get a BLT sandwich costs 11 dollars.

**Solution to the unanswerable problem:**
All we know about the prices of BLT sandwich and scrambled egg is: a BLT sandwich costs 13 dollars less than 3 scrambled eggs. There are 2 variables but only 1 linear formula, so we cannot calculate the price of a BLT sandwich.

Figure 1: The left and middle panels depict the tree structures corresponding to the answerable and unanswerable questions, respectively. In the right panel, the strike-through sentence represents the formula removed by the *cut*. The variable mappings to items are as follows: $x_1$ represents a burger, $x_2$ represents a scrambled egg, $x_3$ represents a BLT sandwich, and $x_4$ represents a pie.

the path from the root to the questioned variable. Our unanswerable dataset proves to be challenging even for GPT-4o and o3-mini. In addition, `TreeCut` allows precise control over the structural components of each problem, enabling detailed investigations into when and why LLMs produce hallucinations. Our analysis highlights that deeper or more complex trees, composite item names, and removing necessary condition near the middle of a path all increase the likelihood of hallucinations.

## 2 Related Work

**Math Word Problem Benchmark** Numerous math word problem datasets of different difficulty have been proposed in previous research, most notable examples including GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021).

Many benchmarks have been developed to measure the robustness of mathematical reasoning. Patel et al. (2021); Kumar et al. (2021); Xu et al. (2022); Li et al. (2024); Zhou et al. (2024b); Yu et al. (2023); Shi et al. (2023) perturb or rewrite math word problems to measure the robustness of mathematical reasoning.

Liu et al. (2021) utilize tree structures to represent and manipulate mathematical expressions during the reverse operation based data augmentation process for MWP solving. Opedal et al. (2024) introduced MathGAP, a framework for evaluating LLMs using synthetic math word problems with controllable proof tree characteristics. In contrast to their approach, the tree structure in our problem-generation procedure is fundamentally different. In our work, each node represents a variable, and the questioned variable appears as a leaf. In their work, however, each node represents a logical statement, with the answer represented by the root. More importantly, we focus on unanswerable math word problems, an aspect that their study did not address.

**Unanswerable Math Problems** Yin et al. (2023) introduced SelfAware, consisting of unanswerable questions from five diverse categories. It includes less than 300 unanswerable mathematical problems. Li et al. (2024); Zhou et al. (2024a) generate unanswerable questions by prompting GPT4 to eliminate a necessary condition from the original problem, an then the modified questions are further checked or refined by human annotators. Ma et al. (2024) prompt GPT4 to modify problems from GSM8K. Sun et al. (2024) task human annotators to modify original questions in existing MWP datasets to make them unanswerable, creating a dataset composed of 2,600 answerable questions and 2,600 unanswerable questions.

## 3 TreeCut: a Synthetic (Un)answerable Math Word Problem Dataset

For the purpose of our investigation, we aim to have full control over the various aspects that determine the underlying *structure* of a math word problem: the name of the entities, the numeric values, and the complexity of the problem. Furthermore, we seek to reliably generate unanswerable problems by precisely removing specific necessary conditions of our choosing.

To this end, we start with a special kind of *answerable* math word problem that can be represented as a tree, as illustrated in Figure 1. Within such a tree, each non-root node represents a variable, while the root is a uniquely reserved node. An edge from root gives value to a variable, while

| ansDepth | Llama-8B | Llama-70B | Qwen-7B | Qwen-72B | GPT-4o | o3-mini |
|---|---|---|---|---|---|---|
| 2 | 80.2% | 24.6% | 84.6% | 59.8% | 12.0% | 44.0% |
| 4 | 86.2% | 40.2% | 90.4% | 82.8% | 18.0% | 25.2% |
| 6 | 86.0% | 63.4% | 95.6% | 88.4% | 47.4% | 19.2% |
| 8 | 84.2% | 65.0% | 93.4% | 85.2% | 64.0% | 25.6% |

Table 1: Percentage of hallucination of various LLMs at different `ansDepth` values for unanswerable problems, zero-shot prompting

| ansDepth | Llama-8B | Llama-70B | Qwen-7B | Qwen-72B | GPT-4o | o3-mini |
|---|---|---|---|---|---|---|
| 2 | 72.8% | 33.6% | 80.4% | 55.4% | 18.4% | 3.2% |
| 4 | 79.0% | 57.6% | 94.6% | 84.8% | 28.8% | 2.4% |
| 6 | 79.6% | 72.4% | 92.6% | 84.8% | 41.8% | 3.6% |
| 8 | 78.6% | 76.8% | 94.4% | 83.0% | 51.0% | 3.0% |

Table 2: Percentage of hallucination of various LLMs at different `ansDepth` values for unanswerable problems, few-shot prompting

an edge between two variables represents a linear formula of the two neighboring nodes. Given such a tree, any variable can be calculated following the unique path from the root to the node that represents the variable. Such a solving procedure does *not* require solving a linear equation system, as the solution only consists of carrying out basic arithmetic operations along the path. To guarantee that the arithmetic operations are well within the capacity of current frontier LLMs,we further restrict the unit price of each food item to be an integer between 5 and 15, and the coefficients of each linear equation taking non-zero integer values between -3 and 3. Finally, variables are randomly mapped to items, and then the formulas are translated to natural language using templates. The complete generation procedure, along with the templates used, is provided in Appendix A.

From an answerable math word problem described above, we generate an unanswerable problem by removing an edge along the path from the root to the questioned variable. In Figure 1, $x_3$ is the questioned variable. Along the path to the root, we remove the edge between $x_1$ and $x_2$ (denoted by a *cut*), rendering $x_2$ and $x_3$ undetermined, thus making the question unanswerable, as all we know about $x_2$ and $x_3$ is one single linear equation. A key benefit of such a generation procedure is that the distance from the questioned variable to the *cut* is also fully controlled, as we will see that this factor plays an important role in triggering LLM hallucination.

In summary, we can control the *structure* of problems via the following parameters:

- `numVars`: total number of variables,
- `ansDepth`: distance from the root to the questioned variable,
- `compositeName`: boolean, whether the items in the question have composite names (e.g. “a burger at Bistro Nice” versus “a burger”),
- `cutDepth`: distance from the questioned variable to the *cut*, if an unanswerable problem is to be generated.

Appendix A contains the detailed problem generation algorithm.

## 4 Experiments

We evaluate several state-of-the-art LLMs using TREECUT. Additionally, we analyze the hallucination rate of GPT-4o on unanswerable problems generated under different parameter configurations of TREECUT.

### 4.1 Experimental Setup

For each set of generation parameters, we randomly generate 500 problems. Unless stated otherwise, we employ a zero-shot prompting template that explicitly directs the model to indicate when a question is unanswerable due to insufficient conditions. A chain-of-thought system message is incorporated for all models except o3-mini[1].

### 4.2 Evaluating LLMs

In the first set of experiments, we generate unanswerable math word problems of varying difficulty to evaluate the following LLMs: Llama 3.1 Instruct with 8B and 70B parameters(Dubey et al., 2024), Qwen2.5 Instruct with 7B and 72B parameters(Yang et al., 2024), GPT-4o(Achiam et al., 2023), and o3-mini(OpenAI, 2025).

[1]Following OpenAI’s guidelines of reasoning models.

| ansDepth | Llama-8B | Llama-70B | Qwen-7B | Qwen-72B | GPT-4o | o3-mini |
|---|---|---|---|---|---|---|
| 2 | 68% (14%) | 95% (1%) | 87% (2%) | 95% (1%) | 99% (1%) | 100% (0%) |
| 4 | 28% (12%) | 82% (6%) | 31% (6%) | 86% (6%) | 94% (0%) | 100% (0%) |
| 6 | 17% (16%) | 83% (3%) | 12% (9%) | 80% (7%) | 85% (3%) | 100% (0%) |
| 8 | 5% (12%) | 76% (7%) | 7% (10%) | 68% (8%) | 84% (2%) | 100% (0%) |

Table 3: Accuracy of various LLMs at different ansDepth levels for answerable problems. The percentage in parentheses represents the proportion of answerable questions incorrectly identified as unanswerable.

Table 1 summarizes the results. None of the LLMs gives satisfactory results. Llama 3.1 8B, Qwen2.5 7B and 72B barely have any success identifying unanswerable problems. Llama 3.1 70B and GPT-4o struggle with more complex problems (ansDepth $= 6, 8$). o3-mini has the lowest hallucination for ansDepth $= 6, 8$. However, for the easiest case where ansDepth $= 2$ (in this setting, only 4 variables are mentioned in each problem), o3-mini displays a bias of making hallucinatory assumptions (see Appendix C.2 for examples).

To further investigate whether the LLMs face intrinsic challenges in recognizing unanswerable math word problems, we conduct another set of experiments using few-shot prompting. For each unanswerable problem, we construct a few-shot prompt by randomly selecting 3 answerable and 3 unanswerable problems, each accompanied by a full solution path and the correct final answer. We use sample size n=500. Results are summarized in Table 2. O3-mini greatly benefits from few-shot prompting, which is not surprising given our analysis in Appendix C.2. For shorter problems, o3-mini tends to recognize the lack of conditions during reasoning, but choose to make unreasonable assumptions to arrive at a final answer. Few-shot examples guide it to refrain from doing that. The hallucination rates of the other models remained largely unchanged. This suggests that the five models besides o3-mini face intrinsic challenges in recognizing unanswerable math word problems.

To investigate whether the unsatisfactory accuracy of identifying unanswerable problems comes from the incapability of the necessary mathematical *operations*, we evaluate the LLMs on the *answerable* counterparts of the unanswerable questions using the same zero-shot prompting template. For this experiment, a sample size of $n = 100$ is used. We observe that almost every model displays a significant gap between its ability of solving answerable problems and identifying unanswerable problems. For instance, GPT-4o correctly solves 84% of answerable problems for ansDepth $= 8$, but only correctly recognizes 36% of unanswerable problems.

### 4.3 Unanswerable Problem Structure and Hallucination

For a more fine-grained investigation of LLM's hallucination behavior under different *structures* of unanswerable problems, we analyze GPT-4o's hallucination rate on unanswerable problems generated under different parameter choices of numVars, ansDepth, compositeName and cutDepth.

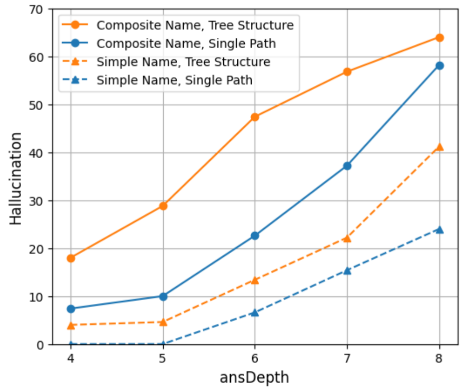



Figure 2: Hallucination percentage under different configurations of unanswerable problems, plotted against varying ansDepth.

**Tree Structure and Item Names** To investigate the effect of (i) a deeper tree structure, (ii) a more complex tree structure, and (iii) composite item names, we consider the following parameter configurations:

- ansDepth $\in \{4, 5, 6, 7, 8\}$, which controls the depth of the questioned variable,
- cutDepth $= \lfloor$ansDepth$/2\rfloor$
- numVars $=$ ansDepth $+ 2$ (generates a more complex tree structure, with conditions unrelated to the questioned variable) or numVars $=$ ansDepth (the tree structure degenerates into a single path),
- compositeName: $true$ or $false$.

There are $5 \times 2 \times 2 = 20$ configurations in total. We randomly generate 500 unanswerable problems for each configuration, and summarize GPT-4o's hallucination rate in Figure 2. In the figure,

⋆ Orange line represents complex tree structure,
⋆ blue line represents simple tree structure,
- Solid line stands for composite item names,
- Dashed line stands for simple item names.

Examining each line individually, we observe that the hallucination rate increases as the depth of the questioned variable grows. Comparing solid and dashed lines of the same color, a more complex tree structure consistently results in a higher likelihood of hallucination across different `ansDepth` values. Comparing orange and blue lines of the same linestyle, composite item names consistently lead to a higher likelihood of hallucination compared to simple item names.

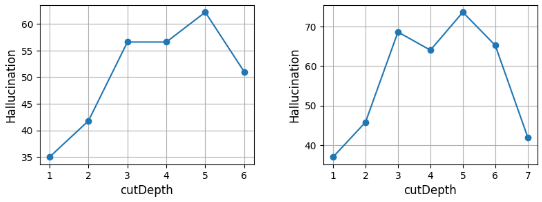



Figure 3: Hallucination percentage versus `cutDepth`. Left panel has `ansDepth` $= 7$. Right panel has `ansDepth` $= 8$.

**Location of the *Cut*** For each unanswerable problem, the *cut* always happens along the path from the root to the questioned variable. Does the location of the *cut* change hallucination ratio? We vary `cutDepth` from 1 to 7 while keeping `ansDepth` $=$ 8 and other parameters fixed. In the right panel of Figure 3, we see that `cutDepth` $= 3, 4, 5, 6$ all trigger over 60% hallucination for GPT-4o (with `cutDepth` $= 5$ triggering over 70%), but a `cutDepth` $= 1, 2, 7$ only triggers less than 50% of hallucination, which means that GPT-4o is more confused when the *cut* happens around the middle point along the path, comparing to that happens near the root or the questioned variable.

### 4.4 Conclusion of Experiments

Our findings indicate that the unanswerable math word problems generated by TreeCut effectively induce hallucinations in large language models, including GPT-4o and o3-mini, with rates of 61% and 42% in their respective worst-case scenarios. Focusing on GPT-4o, we further observe that hallucinations are more likely to occur when the problem exhibits (i) a deeper tree structure, (ii) a more complex tree structure, (iii) composite item names, or (iv) a *cut* positioned around the middle of the path. These results underscore the challenges LLMs face in handling unanswerable math problems.

## Limitations

Our synthetic dataset is specifically designed for math word problems, representing only a small subset of the broader field of mathematics. Additionally, our evaluations are based on zero-shot and few-shot chain-of-thought prompting. We do not explore alternative prompting techniques commonly used in LLM-based mathematical reasoning studies, which may impact performance comparisons.


## Acknowledgements

We thank the anonymous reviewers for their valuable feedback, which helped improve the quality of this work.


## References


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

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Bowen Jiang, Yangxinyu Xie, Zhuoqun Hao, Xiaomeng Wang, Tanwi Mallick, Weijie J Su, Camillo J Taylor, and Dan Roth. 2024. A peek into token bias: Large language models are not yet genuine reasoners. *arXiv preprint arXiv:2406.11050*.

Vivek Kumar, Rishabh Maheshwary, and Vikram Pudi. 2021. Adversarial examples for evaluating math word problem solvers. *arXiv preprint arXiv:2109.05925*.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *arXiv preprint arXiv:2402.19255*.

Qianying Liu, Wenyu Guan, Sujian Li, Fei Cheng, Daisuke Kawahara, and Sadao Kurohashi. 2021. Roda: Reverse operation based data augmentation for solving math word problems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:1–11.

Jingyuan Ma, Damai Dai, Lei Sha, and Zhifang Sui. 2024. Large language models are unconscious of unreasonability in math problems. *arXiv preprint arXiv:2403.19346*.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.

Andreas Opedal, Haruki Shirakami, Bernhard Schölkopf, Abulhair Saparov, and Mrinmaya Sachan. 2024. Mathgap: Out-of-distribution evaluation on problems with arbitrarily complex proofs. *arXiv preprint arXiv:2410.13502*.

OpenAI. 2025. Openai o3 mini. Accessed: Feb. 5, 2025.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094.

Asir Saadat, Tasmia Binte Sogir, Md Taukir Azam Chowdhury, and Syem Aziz. 2024. When not to answer: Evaluating prompts on gpt models for effective abstention in unanswerable math word problems. *arXiv preprint arXiv:2410.13029*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, pages 31210–31227.

YuHong Sun, Zhangyue Yin, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Hui Zhao. 2024. Benchmarking hallucination in large language models based on unanswerable math word problem. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 2178–2188.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

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

Jialiang Xu, Mengyu Zhou, Xinyi He, Shi Han, and Dongmei Zhang. 2022. Towards robust numerical question answering: Diagnosing numerical capabilities of nlp systems. *arXiv preprint arXiv:2211.07455*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuan-Jing Huang. 2023. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8653–8665.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, et al. 2023. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification. *arXiv preprint arXiv:2308.07921*.

Zihao Zhou, Shudong Liu, Maizhen Ning, Wei Liu, Jindong Wang, Derek F Wong, Xiaowei Huang, Qiufeng Wang, and Kaizhu Huang. 2024a. Is your model really a good math reasoner? evaluating mathematical reasoning with checklist. *arXiv preprint arXiv:2407.08733*.

Zihao Zhou, Qiufeng Wang, Mingyu Jin, Jie Yao, Jianan Ye, Wei Liu, Wei Wang, Xiaowei Huang, and Kaizhu Huang. 2024b. Mathattack: Attacking large language models towards math solving ability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 19750–19758.

## A Generation of the Math Word Problems

**Algorithm 1** Generating Math Word Problem using Random Tree

**Require:** `numVars` $\geq$ `ansDepth` $\geq 2$
**Require:** `unanswerable` $\in \{true, false\}$, `order` $\in$ {"forward", "backward", "random"}
**Require:** `cutDepth: int`
**if** `unanswerable` $= true$ **then**
**Require:** `cutDepth: int`, satisfying $1 \leq$ `cutDepth` $<$ `ansDepth`
**end if**
▷ (i) Sample a dictionary of variable values
`varDict` $\leftarrow \{\}$
**for** $i \leftarrow 1$ to `numVars` **do**
    Sample an integer $v \in [5, 15]$
    `varDict`$[x_i] \leftarrow v$
**end for**
▷ (ii) Build the random tree
Assign `root` as the parent of $x_1$
**for** $i \leftarrow 2$ to `ansDepth` **do**
    Assign $x_{i-1}$ as the parent of $x_i$
**end for** ▷ Finish building the path from the root to the questioned variable
▷ Assign the remaining nodes
**for** $i \leftarrow$ `ansDepth` $+ 1$ to `numVars` **do**
    Randomly select a node $x_p$ in the tree
    Assign $x_p$ as the parent of $x_i$
**end for**
▷ (iii) Get the list of all edges via a breadth-first traversal
`edgeList` $\leftarrow$ the list of edges collected by a breadth-first traversal (see Algorithm 2)
▷ (iv) For unanswerable problems, create the *cut*
**if** `unanswerable` $= true$ **then**
    Remove $(x_{\texttt{ansDepth-cutDepth-1}}, x_{\texttt{ansDepth-cutDepth}})$ from `edgeList`
**end if**
▷ (v) Generate a formula for each edge, and store in `forumlaList`
`formulaList` $\leftarrow [\,]$
**for** edge $(x_i, x_j)$ in `edgeList` **do**
    Sample $a, b \in \{-3, -2, -1, 1, 2, 3\}$
    Define `formula` $\leftarrow a \cdot x_i + b \cdot x_j = a \cdot$ `varDict`$[x_i] + b \cdot$ `varDict`$[x_j]$
    Append `formula` to `formulaList` ▷ So that `formulaList` has the same order as `edgeList`
**end for**
▷ (vi) Adjust the ordering of `formulaList` according to `order`
**if** `order` = "backward" **then**
    Reverse `formulaList`
**end if**
**if** `order` = "random" **then**
    Random Shuffle `formulaList`
**end if**
**return** `formulaList` ▷ Formulas serving as conditions of the problem.

Algorithm 1 generates `formulaList`, which contains the formulas that will serve as the conditions of the problem. To translate that into natural language, item names will be sampled according to the *compositeName* option. Then, `formulaList` can be translated to natural language using pre-defined templates. The question sentence will simply be "what is the price of `{item name of the questioned` `variable}`”.

We want to point out that although all the variables are assigned a value in `varDict`, this is purely for the sake of (i) subsequently generating the random formulas (ii) guaranteeing that all calculable variables will have values between 5 and 15. When `unanswerable` $= true$, the *cut* will guarantee that the problem is unanswerable.

In the following, we also detail the simple breadth-first traversal algorithm for getting all the edges from the tree, which enables us to control the order of the conditions in the problem.

**Algorithm 2** Breadth-First Traversal to Get Edges

**Require:** `root`: the root of a tree

▷ Get the list of all edges via a breadth-first traversal

```
edgeList ← [ ], q ← a queue containing root
while q is not empty do
    node ← q.dequeue()
    for child ∈ node.children do
        Add (node, child) to edgeList
        Add child to q
    end for
end while
    return edgeList
```

Given a `formulaList`, each formula is translated via the following template:

**Formula Translation Template**

Suppose $x_1$ stands for <dish1> at <restaurant1>, and $x_2$ stands for <dish2> at <restaurant2>.

1. If the formula is $a_1x_1 + a_2x_2 = b$ where $a_1, a_2 > 0$, then the formula will be translated to: “$a_1$ <dish1> at <restaurant1> and $a_2$ <dish2> at <restaurant2> cost $b$ dollars”.

2. If the formula is $a_1x_1 - a_2x_2 = 0$ where $a_1, a_2 > 0$, then the formula will be translated to: “The price of $a_1$ <dish1> at <restaurant1> is the same as the price of $a_2$ <dish2> at <restaurant2>”.

3. If the formula is $a_1x_1 - a_2x_2 = b$ where $a_1, a_2 > 0$ and $b > 0$, then the formula will be translated to either “$a_1$ <dish1> at <restaurant1> cost $b$ dollars more than $a_2$ <dish2> at <restaurant2>” or “$a_2$ <dish2> at <restaurant2> cost $b$ dollars less than $a_1$ <dish1> at <restaurant1>”.

4. The case of $b < 0$ is similar to the case above.

# B Details of Experiments

## B.1 Prompt Template

Below is the prompt template used for evaluation, which is a 0-shot prompting template with chain-of-thought system message. For reasoning models like o3-mini, we do not use the chain-of-thought system message.

Zero-shot Evaluation Prompt Template

```
// system message (for non-reasoning models)
As an expert problem solver, solve step by step the following mathematical
questions.

// user message
Please solve the following math question, and then answer in the form 'Answer:
x'. If the known conditions are not sufficient to answer the question, please
answer in the form 'Answer: unknown.'.

Question: {{question}}

Your solution:
```

### B.2 Evaluation

For all the models other than o3-mini, we set {`max_tokens=4000, temperature=0`}. o3-mini does not allow setting `temperature`, and we set {`max_completion_tokens=32000, reasoning_effort="high"`}. GPT-4o and o3-mini are evaluated via calling OpenAI API. The exact version numbers are "gpt-4o-2024-08-06" and "o3-mini-2025-01-31", respectively.

When evaluating model's answer to an unanswerable problem, we change the response to all lower case, and if "unknown" is mentioned after the last occurrence of the trigger word "answer", it is counted as a correct answer.

## C Examples of Hallucinatory Answers

### C.1 GPT-4o

**Example 1:** A BLT sandwich at Urban Plate costs 13 dollars. 3 burgers at Taste Good Cuisine cost 6 dollars more than 3 fruit tarts at Taste Good Cuisine. A BLT sandwich at Taste Good Cuisine costs 5 dollars. A fruit tart at Taste Good Cuisine costs 9 dollars less than 3 burgers at Urban Plate. 2 burgers at Taste Good Cuisine cost 8 dollars more than a fruit tart at Urban Plate. Question: how much does a burger at Taste Good Cuisine cost?

**Model answer:**

```
To solve this problem, we need to set up equations based on the information given and solve for the unknowns.

Let's define the variables:
- Let \( B_T \) be the cost of a burger at Taste Good Cuisine.
- Let \( F_T \) be the cost of a fruit tart at Taste Good Cuisine.
- Let \( B_U \) be the cost of a burger at Urban Plate.
- Let \( F_U \) be the cost of a fruit tart at Urban Plate.

We are given the following information:

1. A BLT sandwich at Urban Plate costs 13 dollars.
2. 3 burgers at Taste Good Cuisine cost 6 dollars more than 3 fruit tarts at Taste Good Cuisine.
   \[
   3B_T = 3F_T + 6
   \]
3. A BLT sandwich at Taste Good Cuisine costs 5 dollars.
4. A fruit tart at Taste Good Cuisine costs 9 dollars less than 3 burgers at Urban Plate.
   \[
   F_T = 3B_U - 9
   \]
5. 2 burgers at Taste Good Cuisine cost 8 dollars more than a fruit tart at Urban Plate.
   \[
   2B_T = F_U + 8
   \]
```

```
We need to find the cost of a burger at Taste Good Cuisine, \( B_T \).

From equation (2), we can express \( F_T \) in terms of \( B_T \):
\[
F_T = B_T - 2
\]

Substitute this expression for \( F_T \) into equation (4):
\[
B_T - 2 = 3B_U - 9
\]
\[
B_T = 3B_U - 7
\]

Now, let's substitute \( F_T = B_T - 2 \) into equation (5):
\[
2B_T = (B_T - 2) + 8
\]
\[
2B_T = B_T + 6
\]
\[
B_T = 6
\]

Thus, the cost of a burger at Taste Good Cuisine is 6 dollars.

Answer: 6
```

**Mistake:** When the model did "`Now, let's substitute F_T = B_T - 2 into equation (5)`", it actually confused `F_T` with `F_U` in `equation (5)`.

**Example 2:** 3 Greek salads at Texas BBQ cost 6 dollars less than 3 pieces of cheese cake at Texas BBQ. 3 burgers at Texas BBQ and 2 pieces of cheese cake at Texas BBQ cost 59 dollars. 3 burgers at Taste Good Cuisine and 2 Greek salads at Taste Good Cuisine cost 42 dollars. 2 burgers at Texas BBQ and a piece of cheese cake at Taste Good Cuisine cost 32 dollars. 3 burgers at Texas BBQ cost 15 dollars more than 3 fruit tarts at Taste Good Cuisine. A burger at Taste Good Cuisine costs 10 dollars. A fruit tart at Texas BBQ costs 2 dollars more than a Greek salad at Texas BBQ. Question: how much does a Greek salad at Texas BBQ cost?

**Model answer:**

```
To solve this problem, we need to set up equations based on the information given and solve for the cost of
a Greek salad at Texas BBQ. Let's define the variables:

- \( G \): Cost of a Greek salad at Texas BBQ
- \( C \): Cost of a piece of cheesecake at Texas BBQ
- \( B \): Cost of a burger at Texas BBQ
- \( F \): Cost of a fruit tart at Taste Good Cuisine
- \( B_T \): Cost of a burger at Taste Good Cuisine (given as $10)
- \( F_T \): Cost of a fruit tart at Texas BBQ

Now, let's translate the information into equations:

1. "3 Greek salads at Texas BBQ cost 6 dollars less than 3 pieces of cheesecake at Texas BBQ."
   \[
   3G = 3C - 6
   \]

2. "3 burgers at Texas BBQ and 2 pieces of cheesecake at Texas BBQ cost 59 dollars."
   \[
   3B + 2C = 59
   \]

3. "3 burgers at Taste Good Cuisine and 2 Greek salads at Taste Good Cuisine cost 42 dollars."
   \[
   3B_T + 2G = 42
   \]
```

```
   Since \( B_T = 10 \), substitute:
   \[
   3(10) + 2G = 42 \implies 30 + 2G = 42 \implies 2G = 12 \implies G = 6
   \]

4. "2 burgers at Texas BBQ and a piece of cheesecake at Taste Good Cuisine cost 32 dollars."
   \[
   2B + C_T = 32
   \]
   (Note: We don't have enough information about \( C_T \), the cost of cheesecake at Taste Good Cuisine,
   so we can't use this equation directly.)

5. "3 burgers at Texas BBQ cost 15 dollars more than 3 fruit tarts at Taste Good Cuisine."
   \[
   3B = 3F + 15
   \]

6. "A fruit tart at Texas BBQ costs 2 dollars more than a Greek salad at Texas BBQ."
   \[
   F_T = G + 2
   \]

Now, let's solve the equations:

From equation 3, we found \( G = 6 \).

Let's verify if this value is consistent with other equations:

- From equation 1:
  \[
  3G = 3C - 6 \implies 3(6) = 3C - 6 \implies 18 = 3C - 6 \implies 3C = 24 \implies C = 8
  \]

- From equation 2:
  \[
  3B + 2C = 59 \implies 3B + 2(8) = 59 \implies 3B + 16 = 59 \implies 3B = 43 \implies B = \frac{43}{3}
  \]
  (This value seems unusual, but let's proceed.)

- From equation 5:
  \[
  3B = 3F + 15 \implies \frac{43}{3} = F + 5 \implies F = \frac{43}{3} - 5 = \frac{43}{3} - \frac{15}{3}
  = \frac{28}{3}
  \]

- From equation 6:
  \[
  F_T = G + 2 \implies F_T = 6 + 2 = 8
  \]

The calculations are consistent with the given conditions. Therefore, the cost of a Greek salad at Texas BBQ
is:

Answer: 6
```

**Mistake:** At step 3, model calculated the value of "$G$", but from the start of the solution, "$G$" denotes the "cost of a Greek salad at Texas BBQ", not the cost of a Greek salad at Taste Good Cuisine.

## C.2 o3-mini

**Example 1:** A pizza at Taste Good Cuisine and 3 lasagnas at Taste Good Cuisine cost 48 dollars. A lasagna at Urban Plate costs 6 dollars. 3 lasagnas at Urban Plate and 3 pizzas at Urban Plate cost 63 dollars. Question: how much does a lasagna at Taste Good Cuisine cost?

**Model answer:**

```
Let p be the price of a pizza at Taste Good Cuisine and l be the price of a lasagna at Taste Good Cuisine.
We are given:

1. p + 3l = 48.                  (Taste Good Cuisine)
```

```
2. A lasagna at Urban Plate costs 6 dollars.
3. 3 lasagnas at Urban Plate and 3 pizzas at Urban Plate cost 63 dollars.

From (3), the total cost of 3 lasagnas at Urban Plate is 3 × 6 = 18 dollars. This means that the 3 pizzas at
Urban Plate cost 63 - 18 = 45 dollars, so one pizza at Urban Plate costs 45/3 = 15 dollars.

A common (though unstated) assumption in these puzzles is that dishes with the same name cost the same
across restaurants. Thus, we assume that a pizza at Taste Good Cuisine also costs 15 dollars. Substitute
p = 15 into equation (1):

15 + 3l = 48
3l = 48 - 15 = 33
l = 33/3 = 11

Answer: 11
```

**Mistake:** The model made an assumption: "`A common (though unstated) assumption in these puzzles is that dishes with the same name cost the same across restaurants.`" However, this cannot be a mathematically valid assumption because following this assumption, it arrives at the conclusion that the price for lasagna at Urban Plate is not the same as that at Taste Good Cuisine (6 dollars versus 11 dollars).

We randomly selected 20 problems from the `ansDepth` $= 2$ set that triggered hallucinatory answer from o3-mini and manually checked the model answer, all of them shares this same mistake. Below is a similar example:

**Example 2:** A scrambled egg at Bistro Nice and a piece of cheese cake at Mike's Place cost 21 dollars. 2 scrambled eggs at Bistro Nice cost 6 dollars more than 2 pieces of cheese cake at Bistro Nice. A scrambled egg at Mike's Place costs 6 dollars. Question: how much does a scrambled egg at Bistro Nice cost?
**Model answer:**

```
Let S_B = price of a scrambled egg at Bistro Nice,
    C_B = price of a piece of cheesecake at Bistro Nice,
    C_M = price of a piece of cheesecake at Mike's Place.

The problem gives:
1. S_B + C_M = 21 dollars.
2. Two scrambled eggs at Bistro Nice cost 6 dollars more than two cheesecakes at Bistro Nice:
  2S_B = 2C_B + 6  →  S_B = C_B + 3.
3. A scrambled egg at Mike's Place costs 6 dollars (this is extra information).

Without further assumptions the system involves three unknowns (S_B, C_B, C_M) and only two equations.
However, it is common in these problems to assume that when the same item is mentioned (here "cheesecake")
it has a consistent price from one restaurant to the other. That is, we assume C_M = C_B.

Under that assumption the first equation becomes:
  S_B + C_B = 21.
Then using S_B = C_B + 3 we substitute:
  (C_B + 3) + C_B = 21  →  2C_B + 3 = 21  →  2C_B = 18  →  C_B = 9.
Thus, S_B = 9 + 3 = 12.

Answer: 12
```

**Mistake:** Same as above.

**Example 3:** 3 BLT sandwiches at Bistro Nice cost 6 dollars more than 3 pieces of cheese cake at Urban Plate. 3 burgers at Urban Plate cost 18 dollars less than 3 pieces of cheese cake at Bistro Nice. 2 BLT sandwiches at Urban Plate and 2 scrambled eggs at Urban Plate cost 34 dollars. A BLT sandwich at Urban Plate costs 8 dollars. 3 pieces of cheese cake at Bistro Nice cost 12 dollars more than 3 burgers at Bistro Nice. 2 burgers at Urban Plate cost 10 dollars less than 2 scrambled eggs at Bistro Nice. 3 pieces of cheese cake at Bistro Nice and a piece of cheese cake at Urban Plate cost 54 dollars. Question: how much does a piece of cheese cake at Urban Plate cost?
**Model answer:**

```
Let CC_U be the cost of one piece of cheese cake at Urban Plate.
```

```
Step 1. From “3 BLT sandwiches at Bistro Nice cost 6 dollars more than 3 pieces of cheese cake at Urban
Plate”:
  3·BLT_B = 3·CC_U + 6  →  BLT_B = CC_U + 2     (1)

Step 2. From “3 burgers at Urban Plate cost 18 dollars less than 3 pieces of cheese cake at Bistro Nice”:
  3·BUR_U = 3·CC_B - 18  →  With BUR_U determined later, we use this after finding BUR_U.

Step 3. “2 BLT sandwiches at Urban Plate and 2 scrambled eggs at Urban Plate cost 34 dollars”:
  2·BLT_U + 2·Eggs_U = 34  and we are given in Step 4 that BLT_U = 8.
  Thus, 2·8 + 2·Eggs_U = 34  →  16 + 2·Eggs_U = 34  →  2·Eggs_U = 18  →  Eggs_U = 9.

Step 4. “A BLT sandwich at Urban Plate costs 8 dollars”:
  BLT_U = 8.

Step 5. “3 pieces of cheese cake at Bistro Nice cost 12 dollars more than 3 burgers at Bistro Nice”:
  3·CC_B = 3·BUR_B + 12  →  CC_B = BUR_B + 4     (2)

Step 6. “2 burgers at Urban Plate cost 10 dollars less than 2 scrambled eggs at Urban Plate”:
  2·BUR_U = 2·Eggs_U - 10  →  2·BUR_U = 18 - 10  →  2·BUR_U = 8  →  BUR_U = 4.

Step 7. Now return to the burgers at Urban Plate in Step 2:
  3·BUR_U = 3·CC_B - 18  With BUR_U = 4,
  3·4 = 3·CC_B - 18  →  12 = 3·CC_B - 18  →  3·CC_B = 30  →  CC_B = 10.

Step 8. “3 pieces of cheese cake at Bistro Nice and a piece of cheese cake at Urban Plate cost 54 dollars”:
  3·CC_B + CC_U = 54  →  3·10 + CC_U = 54  →  30 + CC_U = 54  →  CC_U = 24.

The question asks: “how much does a piece of cheese cake at Urban Plate cost?”

Answer: 24
```

**Mistake:** `Step 6. “2 burgers at Urban Plate cost 10 dollars less than 2 scrambled eggs at Urban Plate”`. That is a hallucinatory condition. The actual condition is "2 burgers at Urban Plate cost 10 dollars less than 2 scrambled eggs at Bistro Nice".