The Education Column

BY

Dennis Komm and Thomas Zeume

ETH Zurich, Switzerland and Ruhr University Bochum, Germany

dennis.komm@inf.ethz.ch and thomas.zeume@rub.de

Assessing GPT Performance in a Proof-Based University-Level Course Under Blind Grading

Ming Ding
Department of Computer Science, ETH Zurich
ming.ding@inf.ethz.ch

Federico Soldà
Department of Computer Science, ETH Zurich
federico.solda@inf.ethz.ch

Weixuan Yuan
Department of Computer Science, ETH Zurich
weyuan@inf.ethz.ch

Rasmus Kyng
Department of Computer Science, ETH Zurich
kyng@inf.ethz.ch

Abstract

As large language models (LLMs) advance, their role in higher education, particularly in free-response problem-solving, requires careful examination. This study assesses the performance of GPT-4o and o1-preview under realistic educational conditions in an undergraduate algorithms course. Anonymous GPT-generated solutions to take-home exams were graded by teaching assistants unaware of their origin. Our analysis examines both coarse-grained performance (scores) and fine-grained reasoning quality (error patterns). Results show that GPT-4o consistently struggles, failing to reach the passing threshold, while o1-preview performs significantly better, surpassing the passing score and even exceeding the student median in certain exercises. However, both models exhibit issues with unjustified claims and misleading arguments. These findings highlight the need for robust assessment strategies and AI-aware grading policies in education.

1 Introduction

The emergence of large language models (LLMs) has revolutionized how we interact with artificial intelligence (AI), and garnered significant attention from researchers and the public. While LLMs have demonstrated impressive capabilities, their true level of intelligence and reasoning remains a subject of debate.

The classical Turing Test proposes that a machine demonstrating human-like responses in conversation could be considered intelligent. Over the past few years, substantial efforts have been devoted to evaluating LLMs from various angles [3]. For example, LLMs can generate essays with their quality rated higher than those produced by humans [14]; pass questions involving communication skills, ethics, empathy, and professionalism in a United States Medical Licensing Examination (USMLE) [2]; achieve passing scores on the reading comprehension test of the Program for International Student Assessment (PISA), a global standardized student assessment [31]; and demonstrate strong performance in solving middle school-level math word problems, with multiple LLMs achieving passing scores and some exceeding 90% accuracy [32]. However, existing evaluation protocols may fall short of comprehensively assessing their reasoning and problem-solving capabilities. Most benchmarks involve structured datasets, such as multiple-choice or numeric questions, which may not fully capture the complexities of real-world tasks. While prior research has demonstrated their potential to assist humans or improve responses through prompt engineering [28, 33], there remains a need to evaluate LLMs in scenarios that require independent reasoning and creativity.

The study presented in this article addresses this gap by evaluating GPT models' performance in solving complex free-response questions, specifically proof-based problems from an undergraduate algorithms course. The tasks require not only creative algorithm design but also rigorous justification of correctness—a skill fundamental to human cognitive problem-solving. To ensure unbiased evaluation, we anonymously submitted GPT-generated responses to exam graders, who evaluated them under the assumption that they were written by students. Unlike prior work that focuses on coarse-grained metrics such as pass/fail rates or overall accuracy, we conduct a fine-grained analysis by examining grader feedback and error patterns. This provides deeper insights into LLMs' limitations in proof-based and free-response problem-solving. Beyond assessing performance, our study contributes to the broader discussion on AI's role in education by highlighting both its potential and its constraints. By exploring these aspects, we aim to clarify how AI can reshape learning and assessment practices, as well as the thresholds it must surpass to genuinely emulate undergraduate students.

Recent advancements in LLMs have introduced several competing products, such as OpenAI's ChatGPT [22], Google's Gemini [11], GitHub's Copilot [10], and SciSpace [27]. As these models continue to improve, their progress necessitates

a reevaluation of the role and effectiveness of academic assessments in the era of AI-assisted learning. In this work, we focus on ChatGPT-4o and o1-preview, the latest models from OpenAI available during our experiment period. GPT-4o ("o" for "omni") is a multilingual, multimodal generative pre-trained transformer developed by OpenAI and released in May 2024. In September 2024, OpenAI released o1-preview, the first of a new series of products, with the full version released later. Despite their rapid development, the performance of OpenAI's latest models in realistic educational settings remains largely unexamined. Our study evaluates their effectiveness in solving complex academic problems.

With a particular focus on GPT-4o and o1-preview, our study makes the following key contributions:

1. Our study is conducted in real educational settings with authentic grading conditions, ensuring that the evaluation closely reflects actual academic environments. The experimental design serves as a reference for measuring AI performance in an unbiased and reliable manner. Specifically, we implement *blind grading*, ensuring that the teaching team remains unaware of the experiment and the origin of the submissions. This prevents any potential bias in evaluation. Additionally, we generate solutions using *naive prompts*—without providing course materials or additional guidance—ensuring that GPT does not receive any advantages from instructor input. Different from other benchmarks that are designed to explore the upper limits of LLM problem-solving capabilities, our approach aims to establish a realistic *lower bound* on model effectiveness in educational environments, reflecting how LLMs might perform when used by students in typical, unstructured settings.

2. We use a set of original, highly challenging exercises drawn from take-home exams in an undergraduate algorithms theory course. These problems are designed to be difficult even with full access to course materials, ensuring that the evaluation assesses genuine problem-solving skills rather than mere retrieval-based responses. Our selected exercises are original, which sets us apart from existing popular evaluations on standardized examinations, which often have extensive learning material available online for trainees (including large sets of example questions and answers).

   Moreover, our dataset consists of free-response, proof-based questions that demand multi-stage reasoning. While AI's abilities have been extensively studied in structured tasks like coding [34] and multiple-choice questions [24], there is limited investigation into AI's performance on complex and proof-based problems. Our dataset therefore provides a strong benchmark

for evaluating GPT's capacity to produce correct, logically consistent, and well-structured responses to expert-level questions.

In Section 6, we present a selection of exercises along with their AI-generated solutions and the corresponding grader feedback and comments. The complete dataset is made available in the appendix of the full version of this paper [7].

Our work not only complements existing benchmarks focused on closed-ended or structured problems but also enriches the landscape of high-difficulty evaluations, such as BIG-Bench Hard [30], by providing open-ended, proof-based tasks.

3. We employ expert human grading to ensure a precise and trustworthy evaluation of both student and AI-generated solutions. Although this approach requires substantial human effort, it significantly enhances the reliability of comparisons across students, GPT-4o, and o1-preview. Additionally, our graders provide detailed feedback and comments, allowing for a *fine-grained analysis* of AI errors. By categorizing error patterns into unjustified claims, misleading statements, and mathematical inaccuracies, this qualitative assessment goes beyond numerical scoring, and provides insights into how the models function at a deeper level and highlights specific areas where improvements are needed. This depth of analysis distinguishes our study and offers clearer insights into AI's strengths and limitations in academic problem-solving.

2 Related Work

Large Language Models (LLMs), such as GPT-4, have been extensively evaluated in mathematics and computer science courses to assess their reasoning and problem-solving capabilities. The evaluation of LLMs has been extensively reviewed [3], we focus here on the studies most relevant to our work.

Tests against standard datasets. A comprehensive description of how LLMs are traditionally evaluated can be found, for instance, in the GPT4 Technical Report [21]. In mathematics, the GSM8K dataset [6] contains 8.5K high-quality, linguistically diverse grade-school math word problems designed to evaluate and enhance the mathematical reasoning abilities of language models. Hendrycks et al. proposed the more challenging MATH dataset [13], which includes 12.5K problems spanning a wide range of topics and difficulty levels. Recent LLMs have also been evaluated on a variety of benchmarks: the AIME 2024 benchmark,

which features advanced competition-level mathematics problems [18]; the GPQA Diamond dataset [24], which focuses on numerical and multiple-choice questions; and the MMLU benchmark [12], a suite of multiple-choice questions covering 57 subjects. Additionally, the MathArena platform[1] offers a live leaderboard and curated mathematical challenges for benchmarking LLMs. A recent study by MathArena researchers [23] evaluated LLMs on the 2025 USAMO and found that, despite partial success by Gemini-2.5-Pro, all models struggled with proof generation. Other evaluations include ARC, the AI2 Reasoning Challenge [5]; WinoGrande, an adversarial Winograd schema challenge at scale [26]; HumanEval, which tests functional correctness in code generation from docstrings [4]; DROP, a reading comprehension benchmark requiring discrete reasoning over paragraphs [8]; and HellaSwag [36]; and others. Most recently, Latif et al. [17] evaluated the ChatGPT o1-preview model on established benchmarks. They observed that while o1-preview excelled in structured tasks, it displayed limitations in problem-solving and adaptive reasoning. Notably, it outperformed undergraduate and postgraduate participants in critical thinking, systematic thinking, data literacy, and creative thinking tasks, and surpassed the highest human scores in scientific reasoning.

The use of LLMs in higher education. The application of AI in higher education has also been widely explored in recent years. For example, Nikolić et al. [20] conducted a multi-institutional, multi-disciplinary evaluation of various Generative AI tools, including ChatGPT-3.5, ChatGPT-4, Copilot, Gemini, SciSpace, and Wolfram, across ten engineering subjects in seven Australian universities. The study evaluated AI performance on diverse assessment types. The tasks were rated as Pass, Fail, Component Pass, or Possible Pass, providing a coarse-grained evaluation of AI's educational capabilities. In addition, Susnjak et al. [29] explored how LLMs challenge the integrity of online exams. Their work emphasized multi-step reasoning strategies, guiding LLMs through iterative self-review and refinement to improve their responses. Yeadon et al. [34] evaluated the performance of GPT-3.5 and GPT-4 in university-level physics coding assignments using Python, concluding that AI-generated work closely approaches the quality of university students' work while remaining detectable by human evaluators.

Tests with human comparison. Several studies have directly compared AI and human performance in academic settings. Stribling et al. [28] assess the capability of GPT-4 to answer questions from nine graduate-level final examinations for scientific trainees in the biomedical sciences. They also examine the impact of multiple styles of GPT-4 queries (prompt patterns) and compare results to student performance. Grading has been performed blinded for most examinations.

[1] https://matharena.ai

Kipp [16] tested GPT-4o on 30 unique anatomical multiple-choice questions from German medical licensing exams, where it consistently outperformed medical students across six state exams. In the study of Bayer et al. [1], ChatGPT-4, GPT-4o, and Google Gemini were tested on a physiology exam for veterinary students, with their results compared to those of the students. ChatGPT-4o achieved the highest score with 90% accuracy (18/20 correct answers), followed by Google Gemini at 75% (15/20), and ChatGPT-4 at 60% (12/20), all outperforming the students' average accuracy of 70% (13.99/20). Finally, Richards et al. [25] conducted a dual-anonymous "quality assurance" marking exercise on four end-of-module assessments within a distance-learning computer science curriculum. A total of 90 scripts, either student-submitted or ChatGPT-generated, were blindly graded. All ChatGPT-generated undergraduate scripts received at least a passing grade, whereas none of the ChatGPT-generated postgraduate scripts achieved a passing grade.

High-difficulty datasets. Several recent studies have explored the limitations and capabilities of large language models (LLMs) in proof-based and symbolic reasoning tasks using relatively small, high-difficulty datasets. BIG-Bench Hard [30] focuses on a suite of 23 challenging tasks from BIG-Bench that were found to be beyond the capabilities of current language models. These tasks are those where prior language model evaluations did not outperform the average human rater, making them valuable for stress-testing reasoning abilities. Building on this, Program-Aided Language models (PAL) [9] were evaluated on 13 tasks spanning mathematical, symbolic, and algorithmic reasoning drawn from BIG-Bench Hard and other sources. DeepMind's AlphaProof system[2] evaluated model performance on six problems from the International Mathematical Olympiad (IMO), four of which were successfully solved. Despite the small dataset, this evaluation demonstrated that carefully selected high-difficulty problems can yield meaningful insights into model capabilities—enough to support claims of "silver medal" performance. Similarly, the miniF2F benchmark [35] for the Lean theorem prover contains a test set of 20 formalized IMO problems, with only six problems having fully verified Lean proofs. This benchmark highlights the difficulty of formal reasoning tasks and has become a standard for evaluating autoformalization and proof synthesis systems in low-data, high-complexity regimes.

3 Methodology

3.1 Experiment Design

The study presented in this article aims to assess the effectiveness of LLMs in performing algorithm exercises assigned to university students, using a blinded grading approach to evaluate solutions written by both students and AI.

[2] https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/

Course and examination information. The study was conducted in a third-year undergraduate-level course at ETH Zurich in the Computer Science Department.[3] As a core course in the Bachelor's major "Theoretical Computer Science", the class tends to attract a more specialized group of students, which introduces some selection bias compared to a typical undergraduate cohort. This introductory course on algorithm theory carries 8 ECTS and involves an estimated workload of 200 hours over a 14-week semester. Topics covered include Randomized Search Trees, Point Location, Bootstrapping Algorithms, Linear Programming, Randomized Algebraic Algorithms, and Parallel Algorithms.

The course evaluation comprises two take-home exams featuring free-response proof-based questions.[4] For each take-home exam, students have two weeks to submit solutions, with unrestricted access to all course materials. They are required to propose algorithmic solutions and provide formal proof of correctness and complexity analysis of proposed algorithms. All solutions must be typeset in LaTeX and submitted as PDF.

Since this is a take-home task, students have access to the Internet and potentially AI tools. To ensure academic integrity as much as possible, besides requiring students to cite any external sources for results not covered in the lectures, the course instructors take additional measures in exercise design. Before release, each exercise undergoes a rigorous design process that includes: (1) Non-expert validation – Ensuring that no off-the-shelf solutions are readily available online or in past problem sets. (2) Expert validation – Conducting test-solving sessions among the teaching team to verify the problem's clarity, difficulty, and originality. These measures help maintain exercises' effectiveness while minimizing the risk of students relying on existing solutions.

Data collection and grading process. Each take-home exam consists of four exercises with multiple subquestions. For the first exam, we gathered 134 student submissions and two AI-generated responses, while for the second, we collected 122 student submissions along with two AI-generated responses.

Each exercise was graded independently, and we collected the data per exercise. Grading was conducted by a teaching team member unaware of the submission source (student or AI), ensuring a blind evaluation process. This approach ensures an unbiased comparison between AI-generated and student solutions, making the study reflective of a realistic educational setting. Our grading team consists of PhD students and top-performing bachelor's and master's students (top 10%) from previous years of the course. To ensure transparency, consistency, and to minimize subjectivity in grading, we implemented a rigorous multi-stage process:

[3] https://inf.ethz.ch

[4] The course also includes written in-person exams, which we excluded from evaluation.

Table 1: Prompt patterns.

Types	Prompts
Problem-Solving Request	"Please try to solve the following problem. Note that there are three questions in total. Generate your answer in LaTeX codes. Your solution should be logically consistent and contain necessary steps and explanations toward the final answers. ⟨LaTeX source code of the problem statement⟩" "Now consider the following problem. There are 4 subquestions in total. Please take the hints into consideration and do not forget the instructions about generating your solution in LaTeX codes."
Reference to Lecture Notes	"Please summarize the following Theorem 4.6 to me." ⟨LaTeX source code of the theorem statement⟩
Formatting Request	"Please directly output LaTeX codes so that I can copy." "Can you clearly state your algorithm and analyze it (still in latex codes)?"

graders first drafted a grading scheme, which was reviewed by course lecturers; this was followed by a first-round grading and a second-round review conducted by PhD-level graders. After grading was completed, we revealed our experiment to all graders and asked for their permission to use the results. All graders gave permission.

AI-solution generation. For each exercise, a member of the teaching team generated solutions using both GPT-4o and o1-preview. This person did not participate in any grading. Since the problem statements are well-structured and clearly worded, no prompt engineering was applied. We used only text-based prompts, without incorporating vision inputs or attachments. Additionally, as the course lecture notes were not provided, the LLMs had to rely solely on prior knowledge from their pre-training to solve the problems. However, an exception was made for exercises that explicitly required the use of certain results from the lecture notes. In such cases, we explicitly supplied the relevant content to ensure the model had access to the necessary information and asked the model to summarize the results to make sure it understood correctly. The detailed prompts used in this experiment are provided in Table 1.

For each exercise and for each GPT model, we repeated the query process

three times and selected the best response for submission. We observed that AI-generated solutions exhibit distinct patterns and formatting, making them easily distinguishable from student submissions. To address this, we made *minimal edits* to remove obvious AI-specific artifacts while preserving the content and reasoning of the solutions. These adjustments helped prevent graders from unintentionally identifying AI-generated responses based on formatting alone. More specifically, our edits include:

1. Removing repeated problem statements, as GPT often restates the question before providing an answer;
2. Eliminating excessive headings and subheadings, since GPT frequently structures responses with multiple hierarchical levels;
3. Omitting unnecessary thought processes, as GPT sometimes explicitly outlines its reasoning before presenting the actual solution;
4. Reformatting bullet points into standard paragraphs to make the writing style more natural and human-like.[5]

In our experiment, we found that only minimal edits to AI-generated responses were needed to support effective blind grading. In proof-based mathematical problems, student submissions naturally vary in structure and formatting due to differences in reasoning style and experience. This natural diversity in student submissions reduces the likelihood that AI-generated solutions stand out, even if with slight presentation deviation from typical answers.

4 Results

In this section, we start with a coarse-grained analysis of score distributions, followed by a fine-grained examination of error patterns based on graders' comments. Additionally, we conducted supplementary experiments using exercises from previous years to further validate our findings.

4.1 Score Comparison

Section 4.1 illustrates the score information for eight exercises across two exams, highlighting variability in performance among students and AI models (GPT-4o and o1-preview). The horizontal axis represents the exercises, while the vertical axis indicates the percentage of full scores achieved. Each exercise's score distribution

[5]Exceptions were made when bullet points are a natural way to present the content.

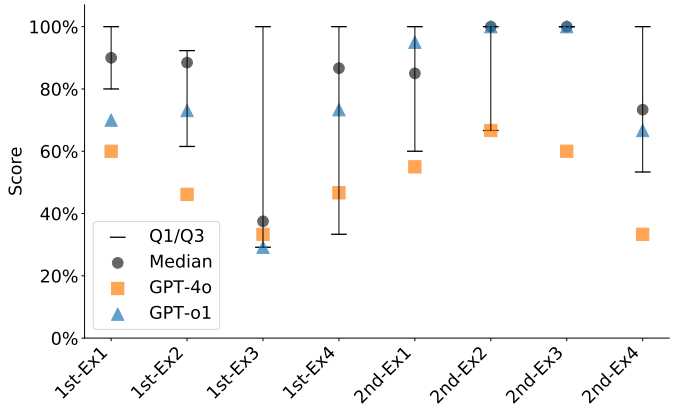

Figure 1: Score per exercise.

is visualized with three key quantile markers: the .25 quantile/Q1 (lower marker), the .5 quantile/median (middle marker), and the .75 quantile/Q3 (upper marker).

The data reveals significant differences in score distributions across exercises, reflecting differences in difficulty levels. For example, in 1st-Ex3, the range between the .25 and .75 quantile is wide, indicating a high level of difficulty. Students' scores are widely dispersed, with some struggling while others excelled. In contrast, for 2nd-Ex3, over 75% of students achieved full points, suggesting that this exercise was relatively easy for most students.

GPT-4o consistently performs below or slightly above the .25 quantile. This indicates significant challenges for GPT-4o in tackling the exercises effectively. Except for 1st-Ex3 where both o1-preview and GPT-4o performed badly, o1-preview constantly outperforms GPT-4o, even by a substantial margin. Notably, in the second exam, o1-preview demonstrates particularly strong performance, exceeding the median and achieving scores close to full marks for 2nd-Ex1, 2nd-Ex2, and 2nd-Ex3.

Exam passing score. Aggregating the scores across all exercises in each exam provides insights into the AI's ability to meet the passing criteria. Table 2 summarizes the results.

For the first exam, where the passing score is set at 50% of the total score across the four exercises, GPT-4o obtained 46%, falling short of the passing threshold and outperforming only 17.65% of students. In contrast, o1-preview achieved 62%, exceeding the passing threshold and outperforming 36.03% of students.

For the second exam, given a relatively easier problem set, the passing score is higher, set at 60% of the total score across the four exercises. GPT-4o obtained 56%, falling short of the passing threshold and outperforming only 6.45% of students. In

Table 2: Exam Passing Information

Exam	Passing	GPT-4o		o1-preview	
	score	score	quantile	score	quantile
First	50	46	17.65%	62	36.03%
Second	60	56	6.45%	92	58.07%

contrast, o1-preview achieved an impressive 92%, exceeding the passing threshold by a large margin and outperforming 58.07% of students.

This analysis highlights the clear superiority of o1-preview over GPT-4o in both of the two exams and demonstrates the variability in their relative performances depending on the difficulty of the exercises and the context of the exam.

4.2 Error Types

Through the above score comparison, we conclude that o1-preview outperforms GPT-4o. In particular, o1-preview has the capability to pass the university-level course with a relatively large margin, while GPT-4o slightly falls below the passing score. In this part, we conduct a fine-grained analysis of each exercise. Based on the graders' feedback and comments, we draw some interesting error patterns and findings, and deep dive into the problem-solving abilities of AI tools.

We categorize the errors identified by graders into three major types:

1. **Unjustified Claims**: Arguments that appear plausible but lack sufficient rigor or proper justification.

2. **Misleading Claims**: Incorrect or invalid arguments presented as valid reasoning.

3. **Mathematical Errors**: These include basic arithmetic or algebraic mistakes, incorrect applications of theorems, failure to address edge cases, misuse of technical terms, and related issues.

The error types identified in solutions generated by GPT-4o and o1-preview are summarized in Table 3. In Section 6 and in the appendix of the full version of this paper [7], the error types and graders' feedback for each AI-generated solution are highlighted in red.

When analyzing the error patterns of GPT-4o and o1-preview, we observe clear differences in their strengths and weaknesses across multiple categories of errors.

Table 3: Error types per exercise.

Exercise	Unjustified Claims		Misleading Claims		Mathematical Errors	
	GPT-4o	o1-preview	GPT-4o	o1-preview	GPT-4o	o1-preview
1st-Ex1	×				×	×
1st-Ex2	×		×		×	×
1st-Ex3	×		×	×		
1st-Ex4	×	×	×		×	×
2nd-Ex1	×	×	×		×	
2nd-Ex2	×		×			
2nd-Ex3			×			
2nd-Ex4	×	×			×	

Unjustified Claims. One of the most prominent issues with GPT-4o is its frequent tendency to make unjustified claims, which occurs in seven out of eight exercises. This suggests that GPT-4o often generates plausible-sounding but ultimately unsupported reasoning. In contrast, o1-preview exhibits this error significantly less frequently, appearing in only three exercises.

A key reason for this discrepancy is that o1-preview produces longer, more structured, and more detailed responses compared to GPT-4o. While verbosity does not necessarily equate to correctness, it appears to help o1-preview provide better justification and avoid making claims without supporting arguments.

Misleading Claims. Another major issue with GPT-4o is its tendency to make misleading claims, which appear significantly more frequently compared to o1-preview. These errors typically involve incorrect arguments presented in a confident manner, making them particularly problematic for grading, as they may appear superficially correct but lack validity upon closer inspection.

However, o1-preview is not entirely free from this issue either. In 1st-Ex3 (Section 6.2), both GPT-4o and o1-preview adopt the same incorrect approach. The significant underperformance compared to students might be AI models' lack of familiarity with the course content compared to the students. Moreover, the inability to reason effectively about geometric concepts may also explain this outcome. Solving 1st-Ex3 requires spatial intuition and geometric reasoning—areas where current LLMs tend to struggle [15, 19].

Mathematical Errors. Both GPT-4o and o1-preview are susceptible to mathematical errors, but GPT-4o commits these errors more frequently and with greater severity. This suggests that GPT-4o may be less precise in its numerical and algebraic reasoning, leading to more fundamental mistakes in problem-solving.

In contrast, o1-preview's mathematical errors tend to be more localized and less impactful on the overall correctness of its solutions. For instance, in subquestion (b) of 1st-Ex2 (Section 6.1), o1-preview misinterprets notation, leading to minor computational errors. However, despite these small mistakes, the overall reasoning and logical approach remain intact, meaning the solution still demonstrates a good understanding of the underlying concepts. This is in stark contrast to GPT-4o, which sometimes fails in a more fundamental way by misapplying mathematical principles altogether.

Other Findings. Course instructors often use the strategy of camouflaging existing problems to prevent students from directly finding solutions online when designing new questions. For example, in 2nd-Ex2 (Section 6.3), students were tasked with analyzing a linear programming technique used in designing an approximation algorithm for the Boolean satisfiability problem (CNF-SAT). To obscure its connection to well-documented materials, the instructor reframed the problem in the context of circuit analysis, substituting key terms with alternative terminology. Specifically, "CNF formula" became "NOT-OR circuit", "literal" was replaced with "input wire", "clause" became "intermediate wire", and "number of satisfied clauses" was termed "output current".

Despite these deliberate modifications, o1-preview successfully identified the underlying mathematical structure and provided a strong solution, whereas GPT-4o struggled. One possible interpretation of this is that o1-preview is more adept at leveraging pre-existing knowledge from publicly available sources, potentially explaining its superior performance on problems with well-documented solutions.

While we do not have conclusive empirical evidence to support this hypothesis, our results represent a small but meaningful step toward understanding the true nature of LLMs' problem-solving abilities: whether they stem from genuine reasoning or from leveraging correlations and latent patterns acquired during training. This remains a fundamental and open research question in the study of language models.

5 Discussion

In this section, we share grader reflections after revealing the GPT experiment to them and discuss potential implications for university educators. We also discuss several limitations of our study.

5.1 Grader Reflections

After completing the experiment, we informed the graders about the study, and several shared their experiences evaluating AI-generated solutions. Surprisingly, except for the grader of 1st-Ex4, who initially expressed suspicion while grading, all other graders stated they could not distinguish AI-generated responses from student submissions, even after being told which ones were AI-generated. This underscores the effectiveness of our minimal editing approach in making AI-written solutions to resemble student writing styles.

More broadly, graders consistently identified "misleading claims" as a major challenge in evaluating AI-generated free-response answers, particularly for complex reasoning tasks. Unlike clearly incorrect responses, misleading claims often appear well-structured and confident, making them deceptively plausible. This forced graders to actively disprove incorrect but superficially logical arguments, which was significantly more time-consuming and cognitively demanding than simply verifying a correct solution.

Implications for university educators. Given the strong performance of LLMs on university-level assignments, we recommend gradually reducing the weight of take-home exams and supplementing them with in-class assessments. In the future, take-home assignments may be more effective when used as *formative* rather than *summative* assessments.

Additionally, it is important to adopt AI-aware evaluation strategies that emphasize transparent and well-justified reasoning. For instance, our analysis of AI-generated responses reveals that LLMs frequently make unjustified or misleading claims. Grading rubrics can be adapted to penalize these types of errors more heavily, helping to distinguish genuine human reasoning from AI-generated artifacts.

The growing capabilities of LLMs also pose new challenges for question design. Simple "camouflaging" of existing problems is no longer sufficient to prevent high-performing LLMs from generating plausible answers. Instructors may need to craft questions that target areas where LLMs still struggle, such as multi-step logical analysis.

5.2 Limitations

Dataset size. Our dataset is relatively small, consisting of 8 exercises with 20 subquestions. Given the limited sample size, the variance of AI-generated solutions may be high, which could introduce some uncertainty in the observed patterns.

However, we emphasize that our dataset is of exceptionally high quality, requiring substantial effort in both exercise design and solution evaluation. The careful

construction of this dataset makes it a valuable resource for evaluating LLMs in an academic setting, and we believe sharing it for further experimentation will contribute meaningfully to the field.

Non-penalized errors. The grading schemes used in this study may not capture all errors made by GPT models. In particular, for "unjustified claims", graders did not always deduct points when such reasoning issues arose, as they were not explicitly penalized under the grading scheme. Instead, graders provided written comments highlighting these issues. As a result, there may be a slight discrepancy between the numerical scores and the fine-grained error analysis.

Despite this, the grading scheme was applied consistently across all student and AI-generated submissions, ensuring that the comparative evaluation between students, GPT-4o, and o1-preview remains reliable. The overall findings still provide meaningful insights into AI performance in solving complex academic problems.

Potential underestimation of LLMs' capabilities. When generating AI solutions, we deliberately used naive prompts—simple, zero-shot inputs without advanced prompt engineering—to avoid human bias. While this approach may not fully activate the reasoning capabilities of LLMs, it aligns with the intended purpose of our benchmark: not to probe the upper limits of LLM performance, but to establish a realistic lower bound on their effectiveness in educational contexts under minimal prompting.

Additionally, since the evaluated tasks were take-home assignments, students may have had access to AI tools and potentially interacted with them using more advanced strategies, such as chain-of-thought (CoT) prompting, which can enhance performance. While this may have inflated student scores, it does not undermine the validity of our assessment of LLM performance. On the contrary, it strengthens our conclusions about o1-preview's capabilities: its ability to achieve comparable or even superior scores in such a setting highlights its problem-solving abilities and underscores the need for reassessing how student performance is evaluated in the presence of powerful LLMs.

6 Selected Exercises

In this part, we include several exercises from our dataset to illustrate and compare the performance of GPT-4o and o1-preview. The complete dataset is available in the appendix of the full version of the paper [7].

6.1 1st-Ex2: Number of Spanning Trees in a Complete Graph

The objective of the exercise is to find the number of distinct spanning trees in a complete graph with n vertices labeled with n distinct labels. An edge between vertices i and j is labeled $\{i, j\}$, and two spanning trees are considered different if they contain any differently labeled edges. For example, on the graph with $n = 3$ labeled vertices, we have 3 distinct spanning trees.

Let $K_n = (V, E)$ be the complete graph with n vertices and suppose that the vertices are labeled $1, 2, \ldots, n$. Let $R \subseteq V : |R| = k$ be a set of k vertices that are fixed (for simplicity, you can think of $R = \{1, 2, \ldots, k\}$). Denote with $T_{n,k}$ the number of (labeled) forests on $\{1, \ldots, n\}$ consisting of k trees whose roots are the vertices in R. Note that $T_{n,k}$ does not depend on R but only on its size and by fixing the roots, the number of possible forests decreases, eg. $T_{3,2} = 2$ as the vertices 1 and 2 cannot be in the same tree (using $R = \{1, 2\}$). For coherence, define also $T_{0,0} = 1$ and $T_{n,0} = 0$ for $n > 0$.

(b) Show that for all $1 \leq k \leq n$,

$$T_{n,k} = \sum_{i=0}^{n-k} \binom{n-k}{i} T_{n-1,k-1+i}.$$

Hint: It might be helpful to use different sets R in recursive cases.

GPT-4o	**o1-preview**
We fix one vertex (say, vertex 1), and consider different ways in which this vertex can be part of a forest: Vertex 1 could either belong to one of the k roots or a subtree of any other root. If vertex 1 belongs to a tree rooted in R (with $\|R\| = k$), we recursively consider the remaining $n-1$ vertices and form $k-1+i$ disjoint trees. **[Unjustified Claims**: Why $k-1+i$? Not explained] The number of ways to choose i vertices from the remaining $n-k$ vertices to add to the forest is given by $\binom{n-k}{i}$.	We aim to establish a recursive relationship for $T_{n,k}$ by considering the role of vertex n in the forests. Let $S = V \setminus R$ be the set of non-root vertices, with $\|S\| = n-k$. We consider how vertex n can be connected in the forest: Since R is fixed and n may or may not be in R, but for consistency, we can assume R does not contain n (if $n \in R$, the argument is similar). **[Mathematical Errors**: No, but you can assume without loss of generality.] Vertex n is not a root (since $n \notin R$), so it must be connected to some other vertex in the forest. We consider all possible ways to connect vertex n: • Connecting to a Root in R: Vertex n connects directly to one of the k roots in R. This does not change the number of trees in the forest. However, since R is fixed, we cannot choose which roots are in R, so we proceed differently.

Continued on the next page...

GPT-4o	o1-preview
Thus, the recurrence relation is: $$T_{n,k} = \sum_{i=0}^{n-k} \binom{n-k}{i} T_{n-1,k-1+i}$$ [**Unjustified Claims**: Why $n-k$? Not explained.]	• Connecting to Non-root Vertices in S: Vertex n connects to one or more [**Mathematical Errors**: Should be "zero or more".] vertices in S. By connecting vertex n to some vertices in S, we might merge their subtrees, possibly reducing the number of trees in the forest. We proceed by considering the number of ways vertex n can be connected to vertices in S: • Choose i vertices from S to connect to n: There are $\binom{n-k}{i}$ [**Mathematical Errors**: Should be $\binom{n-k-1}{i}$.] ways to choose i vertices from S. • Resulting Number of Trees: By connecting n to these i vertices, we potentially reduce the number of trees by i. However, since n is not a root, the total number of roots remains k. So after removing vertex n, the forest consists of $n-1$ vertices and $k-1+i$ trees (since i subtrees may have been merged) [**Mathematical Errors**: i in our formula $k-1+i$ is deg(n) but $i+1$ is deg(n) in your solution.]. Therefore, for each i from 0 to $n-k$, the number of forests is: $$\binom{n-k}{i} T_{n-1,k-1+i}.$$ Summing over all possible i, we obtain: $$T_{n,k} = \sum_{i=0}^{n-k} \binom{n-k}{i} T_{n-1,k-1+i}.$$

6.2 1st-Ex3: Point Location

Given a collection S of n points in the plane and a constant $d > 0$, consider the problem of, given a query point q, finding all the points in S that are at most at distance d from q. Devise a data structure for this problem. In order to get the maximum score, the preprocessing time should be polynomial in the number of points n and if k is the number of points to report, the query should take $O(\log(n)+k)$ operations in expectation.

GPT-4o	o1-preview
[**Misleading Claims**: Wrong approach.] We propose using a k-d tree (k-dimensional tree) to solve this problem. A k-d tree is a binary tree that recursively partitions the points based on their coordinates. The k-d tree is built recursively as follows: • Start with the set of points S and partition it based on one of the coordinates (either x or y). • At the root level, partition the points by the x-coordinate: choose the median of the x-coordinates as the splitting value, and divide the points into two sets: those with x-coordinates smaller than or equal to the median, and those with larger x-coordinates. • Recursively repeat this process for the left and right subtrees. Alternate between splitting by x-coordinate and y-coordinate at successive levels of the tree. **Time complexity for building:** Building the k-d tree takes $O(n \log n)$, because at each level of the recursion, we sort the points and split them, and there are $\log n$ levels in a balanced tree. To query for all points within a distance d from a given point q, we perform a range search on the k-d tree: • Start at the root of the tree and compare the query point q with the splitting coordinate at the current level (either x or y). • If the distance from q to the splitting plane is less than d, recursively search both the left and right subtrees. • If the distance from q to the splitting plane is greater than d, only search the subtree on the side where q lies. • At each leaf node, compute the Euclidean distance between the point and q. If this distance is less than or equal to d, report the point. **Time complexity for querying:** The query time is $O(\log n+k)$, where $\log n$ comes from traversing the tree and k is the number of points reported. [**Unjustified Claims**: Not obvious argument.]	[**Misleading Claims**: Wrong approach.] We propose using a balanced **k-d tree** (k-dimensional tree), which is a binary search tree optimized for organizing points in a multidimensional space—in this case, two dimensions. We build the k-d tree by recursively partitioning the set S as follows: At each level of the tree, we choose a splitting dimension. For a two-dimensional plane, we alternate between the x-coordinate and the y-coordinate. We select the median point along the chosen dimension to split the set. This ensures the tree remains balanced, resulting in a depth of $O(\log n)$. The points are divided into two subsets: those with coordinates less than or equal to the median and those greater than the median, along the splitting dimension. This process continues recursively until each leaf node contains a single point or a small number of points. The preprocessing time to build the k-d tree is $O(n \log n)$, as sorting is required at each level to find medians. **Query Algorithm:** To find all points within distance d from a query point q, we perform a range search on the k-d tree: We start at the root of the tree and compare q's coordinate along the splitting dimension with the node's splitting value. If the absolute difference between q's coordinate and the splitting value is less than or equal to d, both subtrees might contain points within distance d of q. In this case, we recursively search both subtrees. If the difference is greater than d, only one subtree needs to be searched—the one on the same side as q. At each node, if the point stored there is within distance d from q, we add it to the list of points to report. The search continues recursively until all relevant subtrees have been explored. The traversal time through the tree is $O(\log n)$ for a balanced k-d tree. [**Misleading Claims**: Not true if we go through more branches.] The time to report the k points within distance d of q is $O(k)$, since we check each candidate point individually. Therefore, the total expected query time is $O(\log n + k)$.

6.3 2nd-Ex2: NOT-OR Circuits Using Linear Programming

A digital circuit consists of a network of wires connected through logic gates. Each wire carries a signal, which can be either 0 (representing *false*) or 1 (representing *true*). In this exercise, we consider a class of circuits called *NOT-OR circuits*, which consist of three sequential layers.

- **Input Layer**: The circuit has n input wires, where each wire i (for $i = 1, 2, \ldots, n$) carries either a 1 or a 0. The sequence of values on these input wires is called the *input configuration*.
- **Negation Layer**: The n input wires are passed through a layer that outputs $2n$ wires. For each input x_i, this layer produces both x_i and its negation, denoted as not(x_i). These values are referred to as *intermediate wires*.
- **OR Layer**: The intermediate wires are then fed into a layer that outputs m wires, each representing a *OR-gate* value $C_j \in \{0, 1\}$ computed from a subset of the intermediate wires (for $j = 1, 2, \ldots, m$). Each OR-gate value is 1 if at least one of its intermediate wires has value 1. Multiple OR-gates may receive the same intermediate wire as input.

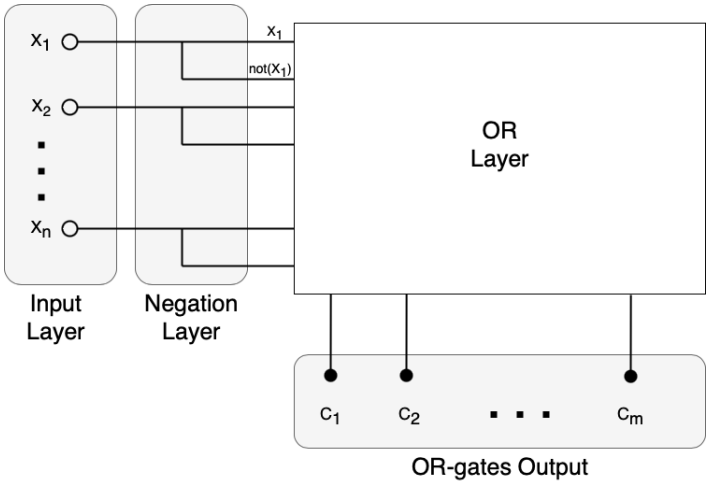

Figure 2: A scheme of a NOT-OR circuit

We consider a NOT-OR circuit where each OR-gate may take as input an arbitrary number of intermediate wires. Each output OR-gate C_j controls a switch that regulates an output current $f_j \geq 0$. The *total output* of the circuit is defined as $\sum_{j=1}^{m} C_j f_j$. Given a circuit and outputs $f_1, \ldots, f_m$, our goal is to find an input configuration that maximizes this total output.

(b) Design a polynomial-time deterministic algorithm that returns an input configuration achieving at least half of the maximum possible output current.
Hint: Consider an input configuration and its negation.

GPT-4o	**o1-preview**
We aim to design a polynomial-time deterministic algorithm that achieves at least half of the maximum possible output current. **Algorithm:** Given a NOT-OR circuit with n input wires, proceed as follows: 1. Compute the output current for the given input configuration x. 2. Compute the output current for the negation of the input configuration not(x). 3. Select the configuration (either x or not(x)) that results in the higher output current. [**Unjustified Claims**: Needs more argumentation, you are basically already assuming the thing you want to show here.] **Justification:** Each OR-gate C_j is connected to the values of the intermediate wires x_i and not(x_i). Therefore, either x or not(x) must contribute at least half of the maximum output current. Since the algorithm evaluates both configurations, it achieves at least half of the maximum possible output current. **Conclusion:** This deterministic algorithm runs in polynomial time and guarantees at least half of the maximum output current. [**Misleading Claims**: Wrong claim.]	**Algorithm:** 1. Compute the total output when all input wires are assigned 0. Let this output be T_0. 2. Compute the total output when all input wires are assigned 1. Let this output be T_1. 3. Choose the input configuration (all zeros or all ones) that yields the higher total output. **Proof of correctness:** Consider any input configuration S and its negation $\bar{S}$ (where each input is flipped). For each OR-gate C_j, at least one of C_j and $\bar{C}_j$ outputs 1 because the inputs to C_j in S are the negation of its inputs in $\bar{S}$. Therefore: $$C_j + \bar{C}_j \geq 1.$$ Let $T_{\max}$ be the maximum possible total output. Then: $$T(S) + T(\bar{S}) = \sum_{j=1}^{m} (C_j + \bar{C}_j) f_j \geq \sum_{j=1}^{m} f_j \geq T_{\max}.$$ Therefore, at least one of $T(S)$ or $T(\bar{S})$ is at least $\frac{1}{2} T_{\max}$.

References

[1] Harun Bayer, Fazilet Gül Ince Araci, and Gülsah Gürkan. Chatgpt-4o, chatgpt-4 and google gemini are compared with students: A study in higher education. *International Journal of Technology in Education and Science*, 8(4):627–644, 2024.

[2] Dana Brin, Vera Sorin, Akhil Vaid, Ali Soroush, Benjamin S Glicksberg, Alexander W Charney, Girish Nadkarni, and Eyal Klang. Comparing chatgpt and gpt-4 performance in usmle soft skill assessments. *Scientific Reports*, 13(1):16492, 2023.

[3] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A Survey on Evaluation of Large Language Models. 15(3):1–45.

[4] 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, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code.

[5] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems.

[7] Ming Ding, Rasmus Kyng, Federico Solda, and Weixuan Yuan. Assessing gpt performance in a proof-based university-level course under blind grading. *arXiv preprint arXiv:2505.13664*, 2025.

[8] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378.

[9] 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.

[10] GitHub. GitHub Copilot: Your AI Pair Programmer, 2024.

[11] Google DeepMind. Gemini AI, 2024.

[12] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*.

[13] 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.

[14] Steffen Herbold, Annette Hautli-Janisz, Ute Heuer, Zlata Kikteva, and Alexander Trautsch. A large-scale comparison of human-written versus chatgpt-generated essays. *Scientific reports*, 13(1):18617, 2023.

[15] Mehran Kazemi, Hamidreza Alvari, Ankit Anand, Jialin Wu, Xi Chen, and Radu Soricut. Geomverse: A systematic evaluation of large models for geometric reasoning. *arXiv preprint arXiv:2312.12241*, 2023.

[16] Markus Kipp. From gpt-3.5 to gpt-4. o: A leap in ai's medical exam performance. *Information*, 15(9):543, 2024.

[17] Ehsan Latif, Yifan Zhou, Shuchen Guo, Lehong Shi, Yizhu Gao, Matthew Nyaaba, Arne Bewerdorff, Xiantong Yang, and Xiaoming Zhai. Can OpenAI o1 outperform humans in higher-order cognitive thinking?

[18] Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic Search-Enhanced Large Reasoning Models.

[19] Spyridon Mouselinos, Henryk Michalewski, and Mateusz Malinowski. Beyond lines and circles: Unveiling the geometric reasoning gap in large language models. *arXiv preprint arXiv:2402.03877*, 2024.

[20] Sasha Nikolic, Carolyn Sandison, Rezwanul Haque, Scott Daniel, Sarah Grundy, Marina Belkina, Sarah Lyden, Ghulam M Hassan, and Peter Neal. Chatgpt, copilot, gemini, scispace and wolfram versus higher education assessments: an updated multi-institutional study of the academic integrity impacts of generative artificial intelligence (genai) on assessment, teaching and learning in engineering. *Australasian journal of engineering education*, 29(2):126–153, 2024.

[21] OpenAI. GPT-4 Technical Report.

[22] OpenAI. ChatGPT: Optimizing Language Models for Dialogue, 2024.

[23] Ivo Petrov, Jasper Dekoninck, Lyuben Baltadzhiev, Maria Drencheva, Kristian Minchev, Mislav Balunović, Nikola Jovanović, and Martin Vechev. Proof or bluff? evaluating llms on 2025 usa math olympiad. *arXiv preprint arXiv:2503.21934*, 2025.

[24] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

[25] Mike Richards, Kevin Waugh, Mark Slaymaker, Marian Petre, John Woodthorpe, and Daniel Gooch. Bob or Bot: Exploring ChatGPT's Answers to University Computer Science Assessment. 24(1):1–32.

[26] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. 64(9):99–106.

[27] SciSpace. SciSpace: AI-Powered Research Platform, 2024.

[28] Daniel Stribling, Yuxing Xia, Maha K Amer, Kiley S Graim, Connie J Mulligan, and Rolf Renne. The model student: Gpt-4 performance on graduate biomedical science exams. *Scientific Reports*, 14(1):5670, 2024.

[29] Teo Susnjak and Timothy R McIntosh. Chatgpt: The end of online exam integrity? *Education Sciences*, 14(6):656, 2024.

[30] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

[31] Esteban Vázquez-Cano, Jose M Ramirez-Hurtado, Jose M Saez-Lopez, and Eloy Lopez-Meneses. Chatgpt: The brightest student in the class. *Thinking Skills and Creativity*, 49:101380, 2023.

[32] Jonathan Vidal. Evaluation of the performance of state-of-the-art large language models (llms) in solving math word problems. *Available at SSRN 4902960*, 2024.

[33] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. 36.

[34] Will Yeadon, Alex Peach, and Craig Testrow. A comparison of human, gpt-3.5, and gpt-4 performance in a university-level coding course. *Scientific Reports*, 14(1):23285, 2024.

[35] Roozbeh Yousefzadeh, Xuenan Cao, and Azim Ospanov. A lean dataset for international math olympiad: Small steps towards writing math proofs for hard problems. *arXiv preprint arXiv:2411.18872*, 2024.

[36] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.