Not All Proofs Are Equal: Evaluating LLM Proof Quality Beyond Correctness

Ivo Petrov [1] **Jasper Dekoninck** [2] **Dimitar I. Dimitrov** [1] **Martin Vechev** [1 2]

GitHub Hugging Face

Abstract

Large language models (LLMs) have become capable mathematical problem-solvers, often producing correct proofs for challenging problems. However, correctness alone is not sufficient: mathematical proofs should also be clear, concise, insightful, and transferable to other problems. While this *proof quality* is subjective and depends on the reader and context, many of its components are concrete and broadly valued. In this work, we identify such components and introduce PROOFRANK, a benchmark curated from challenging mathematical competitions. PROOFRANK evaluates several scalable proxies of proof quality: (i) conciseness, measuring whether proofs avoid unnecessary steps; (ii) computational ease, measuring the extent to which a proof relies on tedious calculations; (iii) cognitive simplicity, measuring how accessible the used proof techniques are; (iv) diversity, measuring how varied a model's proofs for a single problem are; and (v) adaptivity, measuring whether a model can follow a specified proof technique. Across models, we find substantial differences in proof quality that are not captured by correctness-only benchmarks. We also observe significant trade-offs between proof-quality metrics and correctness, suggesting that future evaluations of mathematical reasoning should measure how useful LLM-generated proofs are.

[1]INSAIT, Sofia University "St. Kliment Ohridski" [2]ETH Zurich. Correspondence to: Ivo Petrov <ivo.petrov@insait.ai>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

1. Introduction

Large language models (LLMs) are increasingly used as mathematical collaborators and are now actively assisting with research on open problems (Schmitt, 2025; Ghrist et al., 2024; Raz et al., 2025). A central capability that enables this is theorem proving: an LLM must not only arrive at correct claims, but also justify them rigorously. As a result, proof-based evaluation has become an important component of mathematical reasoning benchmarks, through both human expert evaluation (Petrov et al., 2025a; Dekoninck et al., 2025; Schmitt et al., 2025; Mahdavi et al., 2025a) and automated LLM judges (Luong et al., 2025).

Proof quality Existing proof-writing benchmarks have so far almost entirely focused on proof correctness. However, recent work on AI-assisted mathematics argues that proof value extends beyond formal correctness to include understanding, motivation, and broader mathematical context (Klowden & Tao, 2026). Correct proofs can differ substantially in how clearly they communicate the central idea, how much insight they provide, and whether that insight transfers to related problems or suggests new questions. We refer to these alternative dimensions of a good proof as *proof quality*. Current benchmarks do not capture such distinctions and treat all correct proofs as equivalent.

Scalable proxies for proof quality Evaluating proof quality is challenging because it is inherently subjective: mathematicians may disagree about which proof is more elegant, and the appropriate level of detail can depend on the reader's background. This makes it difficult to define a single absolute score for proof quality. Instead, a scalable benchmark should focus on measurable aspects of proofs that serve as proxies for proof quality. For example, an overreliance on computation is a concrete property that can be evaluated automatically and captures a relevant aspect of proof quality.

This work: PROOFRANK In this work, we introduce PROOFRANK, the first benchmark designed specifically to evaluate these scalable proxies of proof quality in LLM-

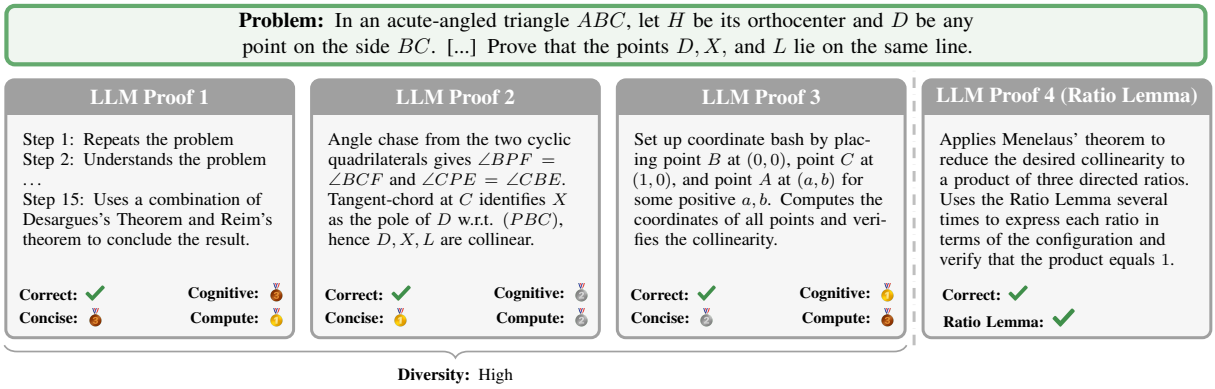

Figure 1. Example of proof quality. Even for the same problem, correct LLM-generated proofs can differ substantially in proof style. This example illustrates all metrics in PROOFRANK, including adaptivity to a specific proof technique (fourth LLM-written proof).

generated proofs. PROOFRANK consists of 382 challenging, high-school level, final-answer competition problems spanning algebra, geometry, combinatorics, calculus, and number theory. We specifically choose to use final-answer problems because they make correctness evaluation substantially more scalable and help avoid known biases in full-proof correctness judgments (Dekoninck et al., 2025).

Metrics in PROOFRANK As shown in Figure 1, we consider five complementary metrics capturing different aspects of proof quality. We group these metrics into two types. *Proof-level* metrics evaluate individual proofs for the same problem. In PROOFRANK, these are conciseness, computational ease, and cognitive simplicity, which capture whether a proof is compact, avoids unnecessary computation, and is easy to follow. *Problem-level* metrics evaluate a model's behavior across multiple attempts to the same problem. In PROOFRANK, these are diversity and adaptivity, which measure whether a model can produce distinct valid approaches and whether it can follow a requested technique.

Pairwise comparisons and correctness Absolute scores are hard to calibrate across problems: what counts as concise for one problem may be verbose for another. To address this, we compare model outputs pairwise for each problem and metric, then aggregate these comparisons into model-level ratings. Additionally, we only compare two outputs when both are correct, meaning they reach the correct final answer and provide a complete justification. This is important because we would otherwise reward the use of very simple but wrong proofs.

Experimental results As shown in Figure 2, we find substantial differences in proof quality across models. GPT-5.4 clearly leads, achieving the highest accuracy and performing best on three of the five quality metrics. In contrast, GEMINI-3.1-PRO performs particularly poorly on conciseness: its solutions are on average 3.5 times more verbose than necessary. We also find that some quality metrics are not correlated with accuracy, suggesting that they capture distinct aspects of proof quality that are not fully aligned with correctness. For instance, QWEN3.5-397B performs better on quality metrics than its low accuracy would suggest, while GLM-5 performs below par on most quality metrics despite achieving reasonably good accuracy.

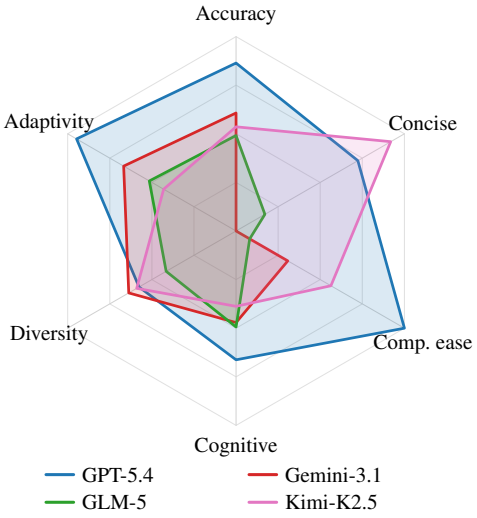

Figure 2. Main results for several models.

Key contributions Our key contributions are:

- Scalable metrics for conciseness, computational ease, cognitive simplicity, diversity, and adaptivity (§3.2).
- PROOFRANK, the first benchmark for evaluating LLM-generated proofs beyond correctness (§3.3).
- An evaluation of frontier LLMs on PROOFRANK, revealing trade-offs in proof quality (§4).

2. Related Work

We briefly discuss related work in mathematical reasoning with LLMs, focusing on proof generation and proof quality.

Proof-generation with LLMs Recent years have seen significant interest in leveraging LLMs for mathematical proof generation, with several works analyzing or benchmarking model performance on this task (Petrov et al., 2025a; Schmitt et al., 2025; Mahdavi et al., 2025a; Frieder et al., 2023a; Guo et al., 2025; Bieganowski et al., 2025; Gonzalez et al., 2026; Peyronnet et al., 2026). These studies show rapid progress and indicate that LLMs can now provide meaningful assistance to professional mathematicians (Schmitt et al., 2025). They also highlight persistent limitations, including sycophancy (Petrov et al., 2025b) and overreliance on brittle pattern matching (Mahdavi et al., 2025a). Since manual proof evaluation is costly, several works have investigated LLMs as judges (Dekoninck et al., 2025; Luong et al., 2025; Ma et al., 2025; Sheng et al., 2025). Related proof-verification benchmarks instead evaluate whether models can grade proofs (Dekoninck et al., 2025; Pandit et al., 2025; Mahdavi et al., 2025b; Ma et al., 2025). These studies show very high agreement with humans, but also reveal systematic biases (Dekoninck et al., 2025; Ma et al., 2025).

Proof quality evaluation While most existing work centers on correctness, recent studies emphasize that correctness alone is not sufficient for LLM-generated mathematics. In particular, Collins et al. (2023) analyzed the relationship between correctness and perceived helpfulness, finding that these metrics are correlated but can diverge substantially. Frieder et al. (2023b) also touch on presentation-related irregularities in their human evaluation of LLM-generated proofs. This fits into a broader shift toward evaluating mathematical models beyond raw accuracy, including reliability (Xue et al., 2025; Kirichenko et al., 2025), interaction-based evaluation (Yue et al., 2026), and structured skill profiles (Zhang et al., 2025). Proof-based benchmarks also often include quantitative error taxonomies (Mahdavi et al., 2025a; Frieder et al., 2023a) and qualitative analyses of proof style (Petrov et al., 2025a; Schmitt et al., 2025), providing insight into issues such as missing structure and hallucinated reasoning. However, these analyses are typically limited in scope and not designed to evaluate multiple dimensions systematically. Our work complements these efforts by quantifying multiple dimensions of proof quality.

LLMs in mathematical education The impact of LLMs on mathematical education has been a topic of active research (Steinbach et al., 2025; Pepin et al., 2025; Turmuzi et al., 2026; Reddig et al., 2025). In educational settings, proof quality is often as important as correctness: explanations must be clear and aligned with the learner's expertise. Recent work highlights risks such as hallucinations and misleading feedback (Steinbach et al., 2025), but also demonstrates the potential of LLMs to generate instructional material and guidance (Reddig et al., 2025).

Formal proof assistants Another line of work integrates LLMs with formal proof assistants like Lean (de Moura & Ullrich, 2021), enabling formally verified proofs (Varambally et al., 2025; Liu et al., 2026; Lin et al., 2025b). Formal benchmarks range from high-school and undergraduate mathematics (Tsoukalas et al., 2024; Azerbayev et al., 2023; Zheng et al., 2022) to research-level or real-world formalization tasks (Ravi et al., 2026b; Jiang et al., 2025; Ravi et al., 2026a; Letson et al., 2026; Taylor et al., 2026). While formal verification reduces the need to judge correctness in natural language, proof quality remains important. For instance, Liu et al. (2026) identify unstructured proofs as a limitation, and Gu et al. (2025) develop techniques to reduce the length of formal proofs generated by LLMs.

LLM-as-a-judge Finally, since many of our metrics use LLMs as judges, we briefly discuss related work in this area. LLM judges are widely applied to open-ended tasks such as instruction-following and question-answering (Zheng et al., 2023; Lin et al., 2025a), as well as mathematical proof generation (Luong et al., 2025; Ma et al., 2025). These works show that LLM judges can achieve high agreement with human evaluators, but may also introduce systematic biases (Ye et al., 2025), including self-preference (Wataoka et al., 2024; Pombal et al., 2026), verbosity bias (Saito et al., 2023), and positional bias (Shi et al., 2025). Our metric design explicitly accounts for these biases.

3. Methodology

In this section, we describe how PROOFRANK evaluates proof quality. We first discuss general considerations that motivate our use of scalable proxies and pairwise comparisons (§3.1). We then define the proof-quality metrics used in the benchmark (§3.2), and finally describe how these metrics are instantiated in PROOFRANK (§3.3).

3.1. Measuring Proof Quality Through Scalable Proxies

Proof quality is subjective Proof quality is inherently subjective. Mathematicians may disagree about the quality of a proof for many reasons, including differences in background and taste. The appropriate level of detail also depends on the reader: a short solution may be ideal for an expert, but unhelpful for a student trying to learn a new concept. For this reason, there is no universally accepted way to determine, in an absolute sense, which model produces the highest-quality proofs.

Scalable proxies Instead, we measure concrete properties that serve as scalable proxies for aspects of proof quality. These proxies are diagnostic rather than prescriptive. For example, a conciseness metric can reveal which model produces shorter proofs, without assuming that every mathematician would always prefer the shortest proof. These measurements are valuable because they are interpretable, scalable, and expose systematic model tendencies.

Unit of comparison Proof-quality proxies can apply to individual proofs or to the models that produce them. Some valuable metrics are not properties of any single proof, but still capture how useful and high-quality a model's proofs are overall. For instance, a model that produces a more diverse set of solutions to the same problem may be more useful because it can provide different insights and make it easier to adapt ideas to related problems. To capture this, we consider two types of metrics: *proof-level* metrics, which compare individual proofs, and *problem-level* metrics, which evaluate a model across multiple solutions to the same problem. Both types are then used to compare models.

Pairwise comparisons Even for scalable proxies, assigning an absolute score to a proof can be difficult. One problem may admit a short conceptual solution, while another may inherently require computations. Thus, a proof that appears computationally heavy for one problem may be unavoidable for another. To avoid calibrating scores across problems, we use pairwise comparisons between two models on the same problem. This makes differences easier to interpret: one can compare which of two proofs is more concise or computationally easier without needing an absolute scale across all problems. These comparisons can then be converted into model rankings using rating systems, which are commonly used in benchmarks (Zheng et al., 2023).

Correctness as a prerequisite A proof can only be of high quality if it is correct. In our experiments, we found that LLMs often produce elegant-looking but incorrect proofs when they fail to solve a problem. Such solutions provide no mathematical value, but can artificially inflate the quality metrics of weaker models. It is therefore important to separate correctness from proof-quality evaluation, so that quality metrics reflect properties of valid proofs rather than artifacts of incomplete or incorrect solutions. We treat correctness as a prerequisite for proof-quality evaluation: proof quality is compared between two models only on problems where both models produced correct solutions.

Final-answer problems Automatic verification of natural-language proof correctness is challenging, and LLM judge biases may propagate into downstream evaluation. For instance, prior work (Dekoninck et al., 2025) finds that LLMs can approach human performance on solution-correctness classification, but also observes limitations, including weaker performance on their own solutions. Moreover, verifying proof correctness often requires running the strongest models on each solution, which can be prohibitively expensive at scale. Thus, we use final-answer problems as proof-writing tasks: models still produce complete written solutions, while correctness can be checked more cheaply.

3.2. Scalable Proxies for Proof Quality

PROOFRANK evaluates five complementary metrics: conciseness, computational ease, cognitive simplicity, diversity, and adaptivity. The first three are proof-level metrics that compare pairs of correct solutions to the same problem, while the last two are problem-level metrics that compare all solutions produced by different models on the same problem. All prompts are given in §F.

Conciseness Verbose proofs can be cumbersome to read and verify, and may contain redundant steps that obscure the main ideas. However, absolute proof length is not a perfect measure of conciseness, since some proofs inherently require more steps than others. To account for this, we measure the *compressibility* of a proof: how much of the text can be removed without changing the logical argument. Specifically, we ask a strong reasoning model to shorten each proof as much as possible while preserving logical equivalence. We then extract valid English words from the original proof p and its shortened version $\tilde{p}$, and define the compressibility ratio $C(p) = |p|/|\tilde{p}|$.

Smaller values indicate that a larger fraction of the original text can be removed without changing the argument. To obtain a pairwise comparison, we compare $C(p_a)$ and $C(p_b)$ for two proofs p_a and p_b of the same problem. If $\frac{|C(p_a)-C(p_b)|}{\max(C(p_a),C(p_b))} < 0.1$, we consider the two proofs to be equally concise, and otherwise we prefer the more concise proof. We use this threshold to avoid treating small differences as meaningful, since such differences can arise from harmless variation in wording. We show that this methodology is robust to the choice of compression model in §B.1.

Computational ease and cognitive simplicity Correct proofs can be difficult to verify for two reasons: they may rely on heavy computation, or they may involve difficult logical dependencies and non-obvious conceptual ideas. We separate these two effects. Computational ease measures how well a proof avoids brute-force algebra, enumeration, case checking, or other calculation-heavy reasoning. Cognitive simplicity measures how easy the argument is to follow, including the complexity of the techniques used and the number of nontrivial connections required. For both metrics, we compare pairs of correct solutions to the same problem and ask an LLM judge (GPT-OSS-120B) which proof is better on the relevant dimension. Details on the validation of these metrics are given in §B.2.

Diversity A model can be more useful if it can produce a variety of correct approaches to the same problem. Such variety can provide different insights and make it easier to adapt ideas to related problems. To measure this, we sample $k = 4$ solutions from each model and cluster the resulting solutions by proof technique.

To perform this clustering, we first ask an LLM (GPT-OSS-120B) to extract a proof summary for each solution, summarizing the main techniques and mathematical tools used. This allows different approaches to be compared without the distraction of heavy computational details or verbose textual information. We then provide all summaries from all models in another LLM call and ask it to cluster them into groups of similar techniques. In particular, we require the LLM to compare the relevant proof style, tools and theorems used, and overall proof structure, while disregarding minor stylistic, notational, and computational differences. The clustering procedure is validated in §B.3.

We measure diversity by quantifying how close the model's distribution over clusters is to uniform. For a model m_i, let $P(C_j \mid m_i)$ be its empirical distribution over technique clusters $C_1, \ldots, C_q$. The entropy of this distribution is

$$H(m_i) = -\sum_{j=1}^{q} P(C_j \mid m_i) \log P(C_j \mid m_i).$$

To obtain a pairwise comparison, we compare $H(m_i)$ and $H(m_{i'})$ for two models m_i and $m_{i'}$ on the same problem, and prefer the model with higher entropy.

Adaptivity Adaptivity measures whether a model can produce a correct proof while following a requested technique. This is an important capability: a student may want to learn how to solve a problem using a particular method, while a researcher may want to explore whether a problem can be solved using different approaches. For each benchmark problem, we first scrape human solutions and use an LLM to summarize these solutions and cluster them by technique, as we do for the diversity metric. Then, for each problem-technique pair (p, t), we prompt the model to solve problem p using the specified technique t. An attempt is counted as successful only if the resulting solution is correct and follows the requested technique. To determine whether the solution follows the requested technique, we give an LLM judge the technique description and ask whether the solution uses it in a non-trivial and logically sound way.

3.3. PROOFRANK

PROOFRANK instantiates these metrics on 382 final-answer problems from established benchmarks, namely MATHARENA (Balunovic et al., 2025) and IMO-ANSWERBENCH (Luong et al., 2025). These tasks cover a wide range of domains, including Algebra, Geometry, Combinatorics, Number Theory, and Calculus. We also collect public solutions for these problems to provide a reference set of human-observed techniques. Although all problems are final-answer problems, most originate from challenging proof-based competitions, and we require a rigorous proof for every evaluated solution. Additional details on problem selection are given in §E.

Correctness verification To instantiate the correctness filter, we use a two-stage correctness heuristic:

- **Final-answer correctness**: Following Luong et al. (2025), we use GPT-OSS-120B to determine whether the model's final answer matches the ground truth.
- **Completeness**: We then ask GPT-OSS-120B to verify that the presented solution is self-contained: each step should follow from the problem statement, a previous step, or a standard mathematical fact. This filters out incomplete or lucky solutions.

A model's accuracy is the fraction of problems where it produces a solution passing both checks.

To validate this filter, we compare it against GPT-5.4 judgments of full proof correctness on 200 randomly sampled solutions, obtaining 93.5% accuracy after manual resolution of discrepancies between GPT-5.4 and our filter. We describe this experiment in more detail in §B.4.

However, this filter introduces a discrepancy between models: weaker models have fewer correct solutions, so their quality metrics are computed on a smaller subset, generally consisting of easier problems. While absolute scores would be affected significantly by this discrepancy, pairwise comparisons are less affected. Two models are only compared on problems where both produced correct solutions, so differences in the difficulty distribution of solved problems do not substantially affect their relative quality scores. In particular, additional comparisons on problems solved only by stronger models primarily affect the relative ratings of those stronger models, rather than giving weaker models credit for problems they did not solve.

Bradley-Terry ranking To convert pairwise outcomes into model rankings, we fit a Bradley-Terry rating model for each pairwise metric. If model m_a is compared against model m_b, the probability that m_a wins is modeled as

$$P(m_a \succ m_b) = \frac{1}{1 + e^{-(\beta_a - \beta_b + \gamma)}},$$

where β_a and β_b are model-specific parameters, and γ is an intercept that captures systematic preference for the first solution, a known issue for LLM judges called position bias (Shi et al., 2025). As in prior work (Zheng et al., 2023), model positions are randomized, and ties are modeled as half a win for each model. We estimate the coefficients using

Table 1. Main model ratings across the different PROOFRANK metrics, measured on correct solutions only, alongside their accuracy rate.

	Conciseness	Computational Ease	Cognitive Simplicity	Diversity	Adaptivity	Accuracy
GPT-5.4	1379	**1615**	**1330**	1261	**1558**	**86.4%**
GEMINI-3.1-PRO	*272*	1046	1177	**1310**	1334	60.7%
KIMI-K2.5-THINK	**1535**	1251	*1110*	1272	1144	53.6%
GLM-5	937	*865*	1195	1132	1212	49.1%
DEEPSEEK-V3.2	1250	1052	1269	1285	1127	47.9%
STEP-3.5-FLASH	1405	1220	1161	1170	1241	41.1%
GPT-OSS-120B	1452	1119	1116	1205	*1079*	34.3%
GROK-4.1-FAST	1470	1012	1163	*1103*	1109	31.4%
GEMINI-3-FLASH	1181	1313	1257	1110	1104	29.8%
QWEN3.5-397B-A17B	1118	1508	1221	1153	1091	*29.3%*

maximum likelihood estimation and report the resulting ratings on an Elo-style scale $r_\alpha = 400\beta_\alpha + 1200$.

Sampling diverse problems and requested techniques Both the diversity and adaptivity metrics require problems that can be solved using multiple techniques. To identify such problems, we collect several human solutions for each problem and cluster them by technique using our LLM-based clustering procedure. For diversity, we require each problem to have at least 3 clusters of human solutions, resulting in 119 problems that are used for this metric. For adaptivity, we first extract every problem-technique pair from the human solutions and then exclude pairs that were already naturally produced by GPT-OSS-120B in its initial solution attempt. This ensures that the requested techniques are not trivial for LLMs. However, it requires care when interpreting the results of GPT-OSS-120B on this metric, since GPT-OSS-120B is disadvantaged by construction. The procedure yields 445 techniques across 237 problems.

4. Experimental Evaluation

We now present our evaluation on PROOFRANK. We first summarize model rankings in §4.1, then analyze conciseness prompting (§4.2), distributional differences between LLM and human solutions (§4.3), and the necessity of correctness filtering in §4.4. We validate the methodology and report further robustness checks in §B and §C.

Evaluated models We evaluate a set of 10 frontier models: GPT-5.4 (OpenAI, 2026), GPT-OSS-120B (OpenAI, 2025), GEMINI-3.1-PRO (Google, 2026), GEMINI-3-FLASH (Google, 2025), QWEN3.5-397B-A17B (Qwen Team, 2026), GLM-5 (GLM, 2026), KIMI-K2.5-THINK (Team, 2026a), GROK-4.1-FAST (xAI, 2025), DEEPSEEK-V3.2 (DeepSeek-AI, 2025), and STEP-3.5-FLASH (Team, 2026b). Additional details on how we ran each LLM are provided in §A.

4.1. Key Takeaways

We present the main results on PROOFRANK in Table 1. Additionally, in Table 2, we report absolute scores for each metric where such scores are measurable, namely for conciseness, diversity, and adaptivity. Confidence intervals of the main results are presented in §C.3. Below, we discuss the key takeaways from these results.

GPT-5.4 ranks highly GPT-5.4 excels in nearly all quality metrics. It ranks first on three of the five, while achieving the highest accuracy by a large margin. Other models show more mixed profiles, and there is no clear second-best model across the different metrics. For instance, while KIMI-K2.5-THINK is most concise, it is worst in terms of cognitive simplicity. At the other end, the worst model in terms of proof quality is GLM-5, with the lowest average ranking across all metrics.

GEMINI-3.1-PRO is extremely verbose GEMINI-3.1-PRO clearly produces the least concise proofs, scoring almost 1300 Elo points below KIMI-K2.5-THINK, the least verbose model. This means that KIMI-K2.5-THINK's solutions are more concise than those by GEMINI-3.1-PRO in over 96% of comparisons. In absolute terms, GEMINI-3.1-PRO's solutions are on average 3.5 times more verbose than necessary, which creates a significant burden for a human reader. Qualitatively, GEMINI-3.1-PRO's solutions almost always restate the goal, define every step in great detail, and often end by restating the result before the final answer.

QWEN3.5-397B and GPT-5.4 dominate computational ease QWEN3.5-397B and GPT-5.4 are the strongest models in terms of computational ease, clearly outperforming all other models. Manual inspection suggests that QWEN3.5-397B's high score is largely explained by its tendency to pursue more elegant ideas, for instance by avoiding brute-force approaches to geometry problems that involve heavy computations, which are common among the other models. However, this comes with a clear tradeoff in correctness, since QWEN3.5-397B has the lowest accuracy

Table 2. Raw scores across the different metrics corresponding to the main results. The raw value of conciseness is listed as compressibility, so lower is better.

	Compressibility	Diversity	Adaptivity
GPT-5.4	1.93	0.32	**77.7%**
GEMINI-3.1-PRO	*3.51*	0.34	52.1%
KIMI-K2.5-THINK	**1.92**	**0.40**	29.2%
GLM-5	2.78	0.17	39.6%
DEEPSEEK-V3.2	2.39	0.38	27.4%
STEP-3.5-FLASH	2.01	0.19	40.9%
GPT-OSS-120B	1.98	0.22	*21.6%*
GROK-4.1-FAST	2.09	*0.15*	25.2%
GEMINI-3-FLASH	2.39	*0.15*	24.5%
QWEN3.5-397B-A17B	2.64	0.22	*22.9%*

among all evaluated models. In contrast, GPT-5.4 achieves a strong balance between correctness and computational ease, suggesting that it can find elegant and direct proofs without sacrificing accuracy.

Accuracy and quality differ substantially Overall, the Spearman rank correlation between accuracy and the quality metrics is relatively low, with rankings on quality metrics differing substantially from rankings by accuracy. For instance, GLM-5 performs below par on all quality metrics despite achieving reasonably good accuracy. The highest correlation between accuracy and any quality metric is 0.84 for adaptivity, which is expected since adaptivity requires correctness as a prerequisite. The lowest correlations are for computational ease and conciseness, both approximately 0, suggesting that these metrics capture aspects of proof quality that are not aligned with correctness.

Cognitive simplicity and diversity are most uniform Cognitive simplicity and diversity are the most uniform metrics across models, with all models scoring relatively close to one another. For instance, on cognitive simplicity, the range of scores is only 200 points, corresponding to a preference rate of only 63% between the best and worst model. In contrast, on conciseness, the range is over 1300 points, corresponding to a preference rate of 96% between the best and worst model.

Differences between absolute and relative metrics The raw scores in Table 2 correlate strongly with the Elo ratings, with Spearman rank correlations of -0.89, 0.93, and 1 for conciseness, diversity, and adaptivity, respectively. The small discrepancy for conciseness arises because models occasionally produce outlier solutions with extremely high compressibility. For instance, GROK-4.1-FAST sometimes duplicates its entire solution. These outliers explain the swap in the ranking of GROK-4.1-FAST and GPT-5.4, which does not have this issue. Similarly, whenever KIMI-K2.5-THINK produces more than one solution, it tends to produce many different ones. This makes its diversity more variable across samples, and explains the swap in the ranking of KIMI-K2.5-THINK and GEMINI-3.1-PRO, which is more consistent across samples.

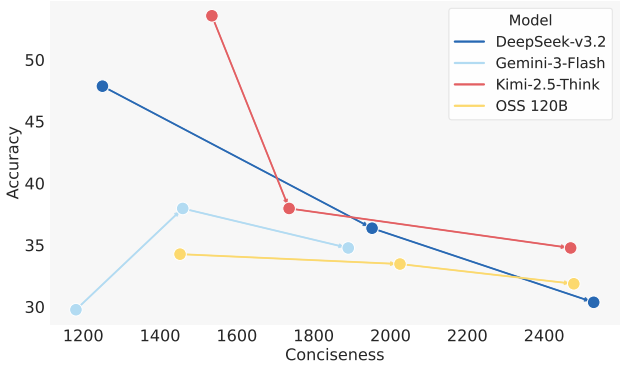

Figure 3. Tradeoff between accuracy and conciseness under different prompting strategies.

4.2. Effect of Prompting on Conciseness

Our main results analyze performance under a simple prompt that does not emphasize any particular style or attribute. This serves as a default-style baseline without explicitly optimizing any quality dimension. However, prompting can significantly affect the style of generated solutions and, consequently, their quality. Conciseness in particular can be strongly influenced by prompting, so we investigate it further in this section.

Prompting strategies We compare the original prompt against two variants. First, we extend it with additional instructions to be as concise as possible. Second, we replace it with instructions that emphasize conciseness even more strongly. The full prompts can be found in §F.

Results We run these ablations on the open-weight models GPT-OSS-120B, DEEPSEEK-V3.2, and KIMI-K2.5-THINK, as well as the proprietary model GEMINI-3-FLASH. In Figure 3, we show how conciseness and accuracy change under the different prompting strategies. As expected, solutions become substantially more concise under the alternative prompts. For two models, this comes at a significant cost in accuracy, with the more concise prompt reducing accuracy by approximately 15% for KIMI-K2.5-THINK and DEEPSEEK-V3.2. In contrast, the accuracy of GPT-OSS-120B and GEMINI-3-FLASH is less affected, and even improves slightly for GEMINI-3-FLASH.

This suggests that conciseness is partially controllable through prompting. This is important for practitioners: some proof-quality proxies can be optimized to match personal preferences, rather than reflecting only fixed model capabilities. Conciseness is especially sensitive to this effect because it is partly a stylistic attribute. However, the ac-

Table 3. Model ratings across the different PROOFRANK metrics (excluding Adaptivity) on all solutions (All). We also show the differences in Elo with respect to the main results.

	Conciseness	**Computational Ease**	**Cognitive Simplicity**	**Diversity**
GPT-5.4	1194 (-155)	1292 (-323)	1122 (-208)	1073 (-188)
GEMINI-3.1-PRO	*499* (+227)	956 (-90)	*1022* (-155)	1142 (-168)
KIMI-K2.5-THINK	1344 (-191)	1154 (-97)	1065 (-45)	1257 (-15)
GLM-5	844 (-93)	*940* (+75)	1242 (+47)	1206 (-74)
DEEPSEEK-V3.2	1180 (-70)	1152 (+100)	1311 (+42)	1242 (-43)
STEP-3.5-FLASH	**1628** (+223)	**1463** (+243)	1265 (+104)	**1365** (+195)
GPT-OSS-120B	1246 (-206)	1045 (-18)	1055 (-61)	1262 (+57)
GROK-4.1-FAST	1590 (+120)	1278 (+266)	1325 (+162)	1157 (+54)
GEMINI-3-FLASH	1328 (+147)	1268 (-45)	1254 (-3)	*1072* (+38)
QWEN3.5-397B-A17B	1146 (+28)	1453 (-55)	**1339** (+118)	1224 (+71)

curacy drops observed for some models show that such prompting is not always a free improvement and needs to be rigorously tested. This motivates our standardized prompt.

4.3. Distributional Differences With Human Solutions

Beyond model rankings, the same clustering pipeline lets us compare LLM-written and human-written solution distributions. We measure the fraction of correct LLM solutions that are not assigned to any human-solution cluster. On average, 31% of correct LLM solutions fall outside the human clusters, with each model deviating by no more than 5% from this value. Similarly, 40% of human solutions are not matched to any LLM solution.

These differences vary by mathematical domain. Many unmatched LLM solutions occur in number theory and geometry. For number theory, models frequently use obscure techniques from the literature that are not present in our collected human-solution set, while for geometry, models often use alternative computational approaches such as complex numbers, trigonometry, or Cartesian coordinates. For example, GEMINI-3-FLASH produces unmatched number theory solutions 45.0% of the time, while GPT-5.4 has a 47.3% unmatched rate in geometry. Importantly, these results should not be interpreted as evidence that LLMs produce new mathematics, since our human solution set is not exhaustive. Rather, they show that LLM-generated solutions often differ from the most common human-written solutions and can reflect different parts of the method distribution.

4.4. Analysis Without Correctness Prerequisite

All preceding analyses compare only correct solutions. To test this design choice, we rerun PROOFRANK without filtering for correctness and include all solutions. We exclude adaptivity, where correctness is required by definition.

Results As shown in Table 3, incorrect solutions have substantial and varying effects across models. The scores of GPT-5.4 are significantly reduced across all metrics, as its correct solutions are now compared against many incorrect, and potentially very simple, solutions from other models. In contrast, the scores of STEP-3.5-FLASH improve dramatically. Qualitatively, we find that its incorrect solutions often provide very high-level but incorrect arguments, leading to strong scores across metrics. Similar trends arise when filtering for a correct answer but allowing incomplete justifications, as we discuss in §C.1. This indicates that filtering for correctness is essential when evaluating proof quality. Otherwise, the results are dominated by the simplicity of incorrect solutions.

5. Limitations

We acknowledge several limitations of our benchmark. First, we do not directly measure proof quality, since there is no objective notion of proof quality that can be easily quantified. Human evaluation is possible, but it is expensive, time-consuming, and may be inconsistent across evaluators. We include a small human evaluation in §B.2 to validate our LLM-as-a-judge components, but we rely primarily on scalable proxies that can be evaluated automatically.

Second, LLM judges are imperfect and may not always align with human judgments of proof quality. They may also exhibit biases or become less reliable as future models develop new proof-writing styles. This is difficult to avoid in scalable evaluations: any automatic proof-quality metric must rely on a heuristic or proxy. We mitigate this issue as much as possible through extensive validation experiments (see §C), which we use to iteratively refine our evaluation components and prompts to make them as reliable and robust as possible. Through these iterations, we also rejected several candidate evaluation methodologies, which we discuss in §D.

Finally, our benchmark focuses on final-answer competition problems, which are a subset of mathematical problems. This allows us to evaluate correctness more easily, but it may limit the generality of our findings to other types of problems, such as proof-writing tasks that require more open-ended reasoning or research-level problems.

6. Conclusion

In this work, we introduced PROOFRANK, a benchmark for assessing the quality of LLM-generated mathematical proofs beyond correctness. We defined a set of metrics that scalably measure proxies for proof quality, including conciseness, computational ease, cognitive simplicity, diversity, and technique adaptivity. Our evaluation reveals distinct strengths and weaknesses among frontier models, as well as nontrivial tradeoffs between accuracy and other proof-quality metrics. This perspective can help users select models whose proof-writing behavior better matches their mathematical goals, and can support future work on steering models toward user-specific proof styles.

Impact Statement

PROOFRANK can support better evaluation of LLMs used for mathematical reasoning by broadening the scope of evaluation beyond correctness to include other important proof-quality dimensions such as conciseness and computational ease. This is useful in almost all mathematical applications, since good proofs should help users understand the underlying mathematical ideas and build intuition. By providing a more nuanced view of model performance, PROOFRANK can help guide users in selecting models that are both accurate and produce high-quality proofs.

Acknowledgements

This research was partially funded by the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure). This project was supported with computational resources provided by Google Cloud Platform (GCP).

References

Azerbayev, Z., Piotrowski, B., Schoelkopf, H., Ayers, E. W., Radev, D., and Avigad, J. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *CoRR*, abs/2302.12433, 2023. doi: 10.48550/ARXIV.2302.12433. URL https://doi.org/10.48550/arXiv.2302.12433.

Balunovic, M., Dekoninck, J., Petrov, I., Jovanovic, N., and Vechev, M. T. Matharena: Evaluating llms on uncontaminated math competitions. *CoRR*, abs/2505.23281, 2025. doi: 10.48550/ARXIV.2505.23281. URL https://doi.org/10.48550/arXiv.2505.23281.

Bieganowski, B., Strzelecki, D., Skiba, R., and Topolewski, M. Putnam-like dataset summary: Llms as mathematical competition contestants. *CoRR*, abs/2509.24827, 2025. doi: 10.48550/ARXIV.2509.24827. URL https://doi.org/10.48550/arXiv.2509.24827.

Collins, K. M., Jiang, A. Q., Frieder, S., Wong, L., Zilka, M., Bhatt, U., Lukasiewicz, T., Wu, Y., Tenenbaum, J. B., Hart, W., Gowers, T., Li, W., Weller, A., and Jamnik, M. Evaluating language models for mathematics through interactions. *CoRR*, abs/2306.01694, 2023. doi: 10.48550/ARXIV.2306.01694. URL https://doi.org/10.48550/arXiv.2306.01694.

de Moura, L. and Ullrich, S. The lean 4 theorem prover and programming language. In *CADE*, volume 12699 of *Lecture Notes in Computer Science*, pp. 625–635. Springer, 2021. doi: 10.1007/978-3-030-79876-5_37. URL https://doi.org/10.1007/978-3-030-79876-5_37.

DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models. *CoRR*, abs/2512.02556, 2025. doi: 10.48550/ARXIV.2512.02556. URL https://doi.org/10.48550/arXiv.2512.02556.

Dekoninck, J., Petrov, I., Minchev, K., Balunovic, M., Vechev, M. T., Marinov, M., Drencheva, M., Konova, L., Shumanov, M., Tsvetkov, K., Drenchev, N., Todorov, L., Nikolova, K., Georgiev, N., Kalinkova, V., and Ismoldayev, M. The open proof corpus: A large-scale study of llm-generated mathematical proofs. *CoRR*, abs/2506.21621, 2025. doi: 10.48550/ARXIV.2506.21621. URL https://doi.org/10.48550/arXiv.2506.21621.

Frieder, S., Pinchetti, L., Chevalier, A., Griffiths, R., Salvatori, T., Lukasiewicz, T., Petersen, P., and Berner, J. Mathematical capabilities of chatgpt. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023a. URL http://papers.nips.cc/paper_files/paper/2023/hash/58168e8a92994655d6da3939e7cc0918-Abstract-Datasets_and_Benchmarks.html.

Frieder, S., Pinchetti, L., Chevalier, A., Griffiths, R., Salvatori, T., Lukasiewicz, T., Petersen, P., and Berner, J. Mathematical capabilities of chatgpt. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023b. URL http://papers.nips.cc/paper_files/paper/2023/hash/58168e8a92994655d6da3939e7cc0918-Abstract-Datasets_and_Benchmarks.html.

Ghrist, R., Gould, J., and Lopez, M. Lattice-valued bottleneck duality, 2024. URL https://arxiv.org/abs/2410.00315.

GLM. GLM-5: from vibe coding to agentic engineering. *CoRR*, abs/2602.15763, 2026. doi: 10.48550/ARXIV.2602.15763. URL https://doi.org/10.48550/arXiv.2602.15763.

Gonzalez, S., Bavandpour, A. A., Ye, P., Zhang, E., Aleksejevs, R., Antic, T., Baron, P., Bhalerao, S., Bhattacharya, S., Burton, Z., Byrne, J., Choi, H., Disha, N. A., Encz, K. I., Fang, Y., George, R. J., Ghorbani, E., Goldfarb, A., Guo, J., Gupta, M., Huber, S., Kanckos, A., Kang, M., Kim, H. J., Lorenzini, D., Lorenzo, L., Mao, T., Marzenta, G., Masuda, A. M., Mauth, L., Mickovic, A., Miniguano-Trujillo, A., Moulin, A., Ni, W., Parry, T., Ren, K., Roodbarani, H., Rundström, M., Saikia, M., Samart, D., Steiner, R., Stewart, C., Thakkar, D., Tse, J., Velona, V., Xiang, Y., Yalçin, S., Yan, J., Zeng, J., Cohan, A., and Liu, Q. C. QEDBENCH: quantifying the alignment gap in automated evaluation of university-level mathematical proofs. *CoRR*, abs/2602.20629, 2026. doi: 10.48550/ARXIV.2602.20629. URL https://doi.org/10.48550/arXiv.2602.20629.

Google. Gemini 3 flash model card. Technical report, Google, December 2025. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf. Accessed: 2026-04-09.

Google. Gemini 3.1 pro model card. Technical report, Google, February 2026. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-1-Pro-Model-Card.pdf. Accessed: 2026-04-09.

Gu, A., Piotrowski, B., Gloeckle, F., Yang, K., and Markosyan, A. H. Proofoptimizer: Training language

models to simplify proofs without human demonstrations. *CoRR*, abs/2510.15700, 2025. doi: 10.48550/ARXIV.2510.15700. URL https://doi.org/10.48550/arXiv.2510.15700.

Guo, D., Liu, J., Fan, Z., He, Z., Li, H., Wang, Y., and Fung, Y. R. Mathematical proof as a litmus test: Revealing failure modes of advanced large reasoning models. *CoRR*, abs/2506.17114, 2025. doi: 10.48550/ARXIV.2506.17114. URL https://doi.org/10.48550/arXiv.2506.17114.

Jiang, J., He, W., Wang, Y., Gao, G., Hu, Y., Wang, J., Guan, N., Wu, P., Dai, C., Xiao, L., and Dong, B. FATE: A formal benchmark series for frontier algebra of multiple difficulty levels. *CoRR*, abs/2511.02872, 2025. doi: 10.48550/ARXIV.2511.02872. URL https://doi.org/10.48550/arXiv.2511.02872.

Kirichenko, P., Ibrahim, M., Chaudhuri, K., and Bell, S. J. Abstentionbench: Reasoning llms fail on unanswerable questions. *CoRR*, abs/2506.09038, 2025. doi: 10.48550/ARXIV.2506.09038. URL https://doi.org/10.48550/arXiv.2506.09038.

Klowden, T. and Tao, T. Mathematical methods and human thought in the age of ai, 2026. URL https://arxiv.org/abs/2603.26524.

Letson, A., Sarra, L., Poiroux, A., Dressler, O., Lezeau, P., Aranha, D., Pu, F., Hill, A., Hidalgo, M. C., Berman, J., Tsoukalas, G., and Taelman, L. Sorrydb: Can AI provers complete real-world lean theorems? *CoRR*, abs/2603.02668, 2026. doi: 10.48550/ARXIV.2603.02668. URL https://doi.org/10.48550/arXiv.2603.02668.

Lin, B. Y., Deng, Y., Chandu, K. R., Ravichander, A., Pyatkin, V., Dziri, N., Bras, R. L., and Choi, Y. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025a. URL https://openreview.net/forum?id=MKEHCx25xp.

Lin, Y., Tang, S., Lyu, B., Yang, Z., Chung, J.-H., Zhao, H., Jiang, L., Geng, Y., Ge, J., Sun, J., Wu, J., Gesi, J., Lu, X., Acuna, D., Yang, K., Lin, H., Choi, Y., Chen, D., Arora, S., and Jin, C. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction, 2025b. URL https://arxiv.org/abs/2508.03613.

Liu, J., Zhou, Z., Zhu, Z., Santos, M. D., He, W., Liu, J., Wang, R., Xie, Y., Zhao, J., Wang, Q., Zhi, L., Li, J., and Li, W. Numina-lean-agent: An open and general agentic reasoning system for formal mathematics, 2026. URL https://arxiv.org/abs/2601.14027.

Luong, T., Hwang, D., Nguyen, H. H., Ghiasi, G., Chervonyi, Y., Seo, I., Kim, J., Bingham, G., Lee, J., Mishra, S., Zhai, A., Hu, C. H., Michalewski, H., Kim, J., Ahn, J., Bae, J., Song, X., Trinh, T. H., Le, Q. V., and Jung, J. Towards robust mathematical reasoning. *CoRR*, abs/2511.01846, 2025. doi: 10.48550/ARXIV.2511.01846. URL https://doi.org/10.48550/arXiv.2511.01846.

Ma, W., Cojocaru, A., Kolhe, N., Louie, B., Sharif, R. S., Zhang, H., Zhuang, V., Zaharia, M., and Min, S. Reliable fine-grained evaluation of natural language math proofs. *CoRR*, abs/2510.13888, 2025. doi: 10.48550/ARXIV.2510.13888. URL https://doi.org/10.48550/arXiv.2510.13888.

Mahdavi, H., Hashemi, A., Daliri, M., Mohammadipour, P., Farhadi, A., Malek, S., Yazdanifard, Y., Khasahmadi, A., and Honavar, V. G. Brains vs. bytes: Evaluating LLM proficiency in olympiad mathematics. *CoRR*, abs/2504.01995, 2025a. doi: 10.48550/ARXIV.2504.01995. URL https://doi.org/10.48550/arXiv.2504.01995.

Mahdavi, H., Mahdavinia, P., Malek, S., Mohammadipour, P., Hashemi, A., Daliri, M., Farhadi, A., Khasahmadi, A., Mireshghallah, N., and Honavar, V. G. Refgrader: Automated grading of mathematical competition proofs using agentic workflows. *CoRR*, abs/2510.09021, 2025b. doi: 10.48550/ARXIV.2510.09021. URL https://doi.org/10.48550/arXiv.2510.09021.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. *CoRR*, abs/2508.10925, 2025. doi: 10.48550/ARXIV.2508.10925. URL https://doi.org/10.48550/arXiv.2508.10925.

OpenAI. Gpt-5.4 thinking system card. Technical report, OpenAI, March 2026. URL https://openai.com/index/gpt-5-4-thinking-system-card/. Accessed: 2026-04-09.

Pandit, S., Xu, A., Nguyen, X., Ming, Y., Xiong, C., and Joty, S. Hard2verify: A step-level verification benchmark for open-ended frontier math. *CoRR*, abs/2510.13744, 2025. doi: 10.48550/ARXIV.2510.13744. URL https://doi.org/10.48550/arXiv.2510.13744.

Pepin, B., Buchholtz, N., and Salinas-Hernández, U. A scoping survey of chatgpt in mathematics education. *Digital Experiences in Mathematics Education*, 11:9–41, 2025. doi: 10.1007/s40751-025-00172-1. URL https://doi.org/10.1007/s40751-025-00172-1.

Petrov, I., Dekoninck, J., Baltadzhiev, L., Drencheva, M., Minchev, K., Balunovic, M., Jovanovic, N., and Vechev, M. T. Proof or bluff? evaluating llms on 2025 USA

math olympiad. *CoRR*, abs/2503.21934, 2025a. doi: 10.48550/ARXIV.2503.21934. URL https://doi.org/10.48550/arXiv.2503.21934.

Petrov, I., Dekoninck, J., and Vechev, M. T. Brokenmath: A benchmark for sycophancy in theorem proving with llms. *CoRR*, abs/2510.04721, 2025b. doi: 10.48550/ARXIV.2510.04721. URL https://doi.org/10.48550/arXiv.2510.04721.

Peyronnet, A., Gloeckle, F., and Hayat, A. Lemmabench: A live, research-level benchmark to evaluate LLM capabilities in mathematics. *CoRR*, abs/2602.24173, 2026. doi: 10.48550/ARXIV.2602.24173. URL https://doi.org/10.48550/arXiv.2602.24173.

Pombal, J., Rei, R., and Martins, A. F. T. Self-preference bias in rubric-based evaluation of large language models. *CoRR*, abs/2604.06996, 2026. doi: 10.48550/ARXIV.2604.06996. URL https://doi.org/10.48550/arXiv.2604.06996.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Ravi, N., Nashold, L., and Krishnan, R. Proofbench: Can models write math proofs that are formally verified? Vals AI, January 2026a. URL https://www.vals.ai/benchmarks/proof_bench. Accessed: 2026-04-29.

Ravi, N., Ying, K., Nesterov, V., Krishnan, R., Uskuplu, E., Xia, B., Aswedige, J., and Nashold, L. Formalproofbench: Can models write graduate level math proofs that are formally verified? *CoRR*, abs/2603.26996, 2026b. doi: 10.48550/ARXIV.2603.26996. URL https://doi.org/10.48550/arXiv.2603.26996.

Raz, T., Shalyt, M., Leibtag, E., Kalisch, R., Weinbaum, S., Hadad, Y., and Kaminer, I. From euler to ai: Unifying formulas for mathematical constants, 2025. URL https://arxiv.org/abs/2502.17533.

Reddig, J. M., Arora, A., and MacLellan, C. J. Generating in-context, personalized feedback for intelligent tutors with large language models. *Int. J. Artif. Intell. Educ.*, 35 (6):3459–3500, 2025. doi: 10.1007/S40593-025-00505-6. URL https://doi.org/10.1007/s40593-025-00505-6.

Saito, K., Wachi, A., Wataoka, K., and Akimoto, Y. Verbosity bias in preference labeling by large language models. *CoRR*, abs/2310.10076, 2023. doi: 10.48550/ARXIV.2310.10076. URL https://doi.org/10.48550/arXiv.2310.10076.

Schmitt, J. Extremal descendant integrals on moduli spaces of curves: An inequality discovered and proved in collaboration with ai, 2025. URL https://arxiv.org/abs/2512.14575.

Schmitt, J., Bérczi, G., Dekoninck, J., Feusi, J., Gehrunger, T., Appenzeller, R., Bryan, J., Canova, N., de Wolff, T., Gaia, F., van Garrel, M., Hashemi, B., Holmes, D., Lopez, A. I., Jaeck, V., Jørgensen, M., Kelk, S., Kuhlmann, S., Kurpisz, A., Meroni, C., Metzler, I., Möller, M., Muñoz-Echániz, S., Nowak, R., Oberdieck, G., Platt, D., Possamaï, D., Ribeiro, G., Galan, R. S., Sun, Z., Teichmann, J., Thomas, R. P., and Vial, C. Improofbench: Benchmarking AI on research-level mathematical proof generation. *CoRR*, abs/2509.26076, 2025. doi: 10.48550/ARXIV.2509.26076. URL https://doi.org/10.48550/arXiv.2509.26076.

Sheng, J., Lyu, L., Jin, J., Xia, T., Gu, A., Zou, J., and Lu, P. Solving inequality proofs with large language models. *CoRR*, abs/2506.07927, 2025. doi: 10.48550/ARXIV.2506.07927. URL https://doi.org/10.48550/arXiv.2506.07927.

Shi, L., Ma, C., Liang, W., Diao, X., Ma, W., and Vosoughi, S. Judging the judges: A systematic study of position bias in llm-as-a-judge. In Inui, K., Sakti, S., Wang, H., Wong, D. F., Bhattacharyya, P., Banerjee, B., Ekbal, A., Chakraborty, T., and Singh, D. P. (eds.), *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, IJCNLP-AACL 2025, Mumbai, India, December 20-24, 2025*, pp. 292–314. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.ijcnlp-long.18/.

Steinbach, M., Bhandari, S., Meyer, J., and Pardos, Z. A. When llms hallucinate: Examining the effects of erroneous feedback in math tutoring systems. In Mills, C., Alexandron, G., Taibi, D., Bosco, G. L., and Paquette, L. (eds.), *Proceedings of the 18th International Conference on Educational Data Mining, EDM 2025, Palermo, Italy, July 20-23, 2025*. International Educational Data Mining Society, 2025. URL https://educationaldatamining.org/2025.EDM.doctoral-consortium-papers.254/index.html.

Taylor, A. K., Zhang, J., Ji, E., Sahai, V., Deng, H., Chen, Y., Yuan, Y., Wu, D., Gu, J., Chang, K., Peng, N., Sahai, A., and Wang, W. Taobench: Do automated theorem prover llms generalize beyond mathlib? *CoRR*, abs/2603.12744, 2026. doi: 10.48550/ARXIV.2603.12744. URL https://doi.org/10.48550/arXiv.2603.12744.

Team, K. Kimi K2.5: visual agentic intelligence. *CoRR*, abs/2602.02276, 2026a. doi: 10.48550/ARXIV.2602.02276. URL https://doi.org/10.48550/arXiv.2602.02276.

Team, S. Step 3.5 flash: Open frontier-level intelligence with 11b active parameters. *CoRR*, abs/2602.10604, 2026b. doi: 10.48550/ARXIV.2602.10604. URL https://doi.org/10.48550/arXiv.2602.10604.

Tsoukalas, G., Lee, J., Jennings, J., Xin, J., Ding, M., Jennings, M., Thakur, A., and Chaudhuri, S. Putnambench: Evaluating neural theorem-provers on the putnam mathematical competition. In Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J. M., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/1582eaf9e0cf349e1e5a6ee453100aa1-Abstract-Datasets_and_Benchmarks_Track.html.

Turmuzi, M., Azmi, S., and Kertiyani, N. M. I. Chatgpt in school mathematics education: A systematic review of opportunities, challenges, and pedagogical implications. *Teaching and Teacher Education*, 170:105286, 2026. doi: 10.1016/j.tate.2025.105286. URL https://doi.org/10.1016/j.tate.2025.105286.

Varambally, S., Voice, T., Sun, Y., Chen, Z., Yu, R., and Ye, K. Hilbert: Recursively building formal proofs with informal reasoning. *CoRR*, abs/2509.22819, 2025. doi: 10.48550/ARXIV.2509.22819. URL https://doi.org/10.48550/arXiv.2509.22819.

Wataoka, K., Takahashi, T., and Ri, R. Self-preference bias in llm-as-a-judge. *CoRR*, abs/2410.21819, 2024. doi: 10.48550/ARXIV.2410.21819. URL https://doi.org/10.48550/arXiv.2410.21819.

xAI. Grok 4.1 fast and agent tools api. *xAI News*, November 2025. URL https://x.ai/news/grok-4-1-fast. Accessed: 2026-04-09.

Xue, B., Zhu, Q., Wang, R., Wang, S., Wang, H., Mi, F., Wang, Y., Shang, L., Liu, Q., and Wong, K. Reliablemath: Benchmark of reliable mathematical reasoning on large language models. *CoRR*, abs/2507.03133, 2025. doi: 10.48550/ARXIV.2507.03133. URL https://doi.org/10.48550/arXiv.2507.03133.

Ye, J., Wang, Y., Huang, Y., Chen, D., Zhang, Q., Moniz, N., Gao, T., Geyer, W., Huang, C., Chen, P., Chawla, N. V., and Zhang, X. Justice or prejudice? quantifying biases in llm-as-a-judge. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=3GTtZFiajM.

Yue, B., Zhu, Z., Zhang, Y., Feng, J., Yang, H., and Wang, M. Interactive benchmarks. *CoRR*, abs/2603.04737, 2026. doi: 10.48550/ARXIV.2603.04737. URL https://doi.org/10.48550/arXiv.2603.04737.

Zhang, Y., Zhang, J., Ren, Q., Saffat, T., Liu, X., Yang, Z., Zhu, B., and Ma, Y. GAUSS: benchmarking structured mathematical skills for large language models. *CoRR*, abs/2509.18122, 2025. doi: 10.48550/ARXIV.2509.18122. URL https://doi.org/10.48550/arXiv.2509.18122.

Zheng, K., Han, J. M., and Polu, S. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=9ZPegFuFTFv.

Zheng, L., Chiang, W., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging llm-as-a-judge with mt-bench and chatbot arena. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

A. Additional Experimental Details

We provide additional experimental details that are not included in the main text to ensure full transparency and reproducibility.

A.1. Model Details and Inference

Hyperparameters All models were run using the hyperparameters recommended by their respective providers. We provide the exact configurations used for each model in our code. Unless otherwise specified, models were run with their maximum available reasoning budget. We retried a generation only when the corresponding request exceeded a timeout of 150 minutes. For GPT-5.4, we used the `background=True` parameter, which was necessary to avoid connection failures during long requests.

Parsing and Format Adherence For prompts requiring a structured output format, such as diversity clustering or core-idea generation, we implemented task-specific parsers to extract the required fields from model responses. These parsers were also used to identify formatting errors and internal inconsistencies in the outputs. When a response did not conform to the expected format, we retried the prompt up to 10 times. If all retry attempts failed, the corresponding instance was marked as a formatting failure and excluded from the affected dataset. In practice, such failures were rare, occurring in less than 0.5% of runs in each experiment.

A.2. Computational Requirements

For larger and closed-source LLMs, we used the providers' API services, as well as the OpenRouter API. To improve throughput, we issued between 32 and 128 concurrent API requests, depending on provider rate limits and model availability.

All local inference experiments, including those using GPT-OSS-120B and QWEN3.5-397B-A17B, were run on a single node with 8 NVIDIA H200 GPUs, each with 144 GB of VRAM. These runs used at most 144 GB of system RAM and 32 CPU cores. In practice, most experiments could be run on 1 to 4 H200 GPUs, although with reduced throughput.

B. Validation of Methodology and Metrics

Because our evaluation targets subjective and nuanced aspects of mathematical proof generation, we perform additional validation experiments to assess the reliability of our methodology and the robustness of our metrics. In particular, we focus on components that use an LLM-as-a-judge: rephrasing-based conciseness measurement, pairwise preference judgment, clustering consistency, and correctness verification.

B.1. Effect of Rephraser on Conciseness Measurement

Since our conciseness metric relies on an LLM rephraser to measure the compressibility of a solution, we test whether these measurements are robust to the choice of rephraser. Here, we compare our main verifier, GPT-OSS-120B, with GROK-4.1-FAST.

We find that GROK-4.1-FAST tends to produce shorter rephrasings, and carries over this capability to this task by compressing each solution more aggressively. However, the relative comparisons between models remain consistent. As shown in Figure 4, the average shift in Elo ratings is only 4.1%. Although some model scores change, the overall ranking trends are preserved, with few rank inversions.

We use GPT-OSS-120B in our main experiments because it is a more reliable rephraser, producing semantically valid rephrasings in 94.1% of cases. In contrast, GROK-4.1-FAST achieves only 88.1%. We describe this experiment in more detail in §C.2.

B.2. Human-driven Validation

To assess the reliability of our methodology, we asked three authors with sufficient mathematical background to compare pairs of solutions along three primary metrics: *conciseness*, *computational ease*, and *cognitive simplicity*. We also considered two additional qualitative dimensions: *insightfulness* and *elegance*. Insightfulness measures whether a solution introduces transferable ideas or techniques, while elegance captures whether the proof uses a simple, creative, or unexpectedly efficient

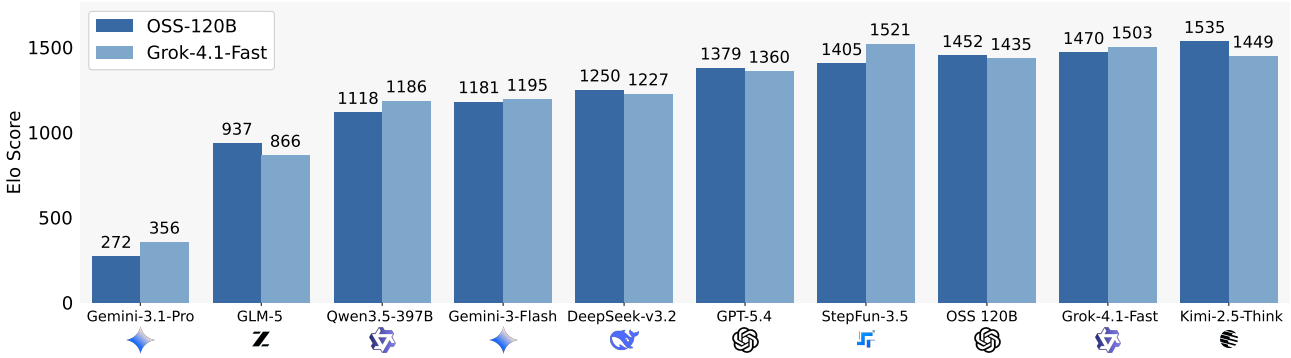

Figure 4. Elo changes when only considering valid rephrasings for the conciseness metric.

Table 4. Average inter-annotator agreement from our human annotation study.

Conciseness	Computational Ease	Cognitive Simplicity	Insightfulness	Elegance
57.6%	83.9%	80.1%	87.0%	89.0%

argument. The instructions given to annotators are provided in §F.1.

We collected 90 pairwise judgments for each dimension. Forty-two of these judgments were independently annotated by two annotators. To summarize inter-annotator consistency, we use a graded agreement score: 100% for identical preferences, 50% when one annotator expressed no preference and the other preferred one solution, and 0% for opposing preferences. The results are shown in Table 4. Overall, the results show good agreement for 4 out of the 5 dimensions.

Inter-annotator agreement Agreement is highest for *insightfulness* and *elegance*. Although these dimensions are subjective, the high agreement suggests that annotators often share similar preferences, at least within the context of our examples. However, all three authors know each other and discussed this project together, so their judgments may be more aligned than those of independent annotators.

Annotators also achieve agreement above 80% for *computational ease* and *cognitive simplicity*, suggesting that these criteria can be applied consistently in pairwise comparisons.

The main exception is *conciseness*. Annotators noted that different solutions were often verbose in different ways, so judgments depended on which redundant parts they emphasized. After annotation, the annotators agreed that they had often tried to approximate our compressibility metric. However, doing so reliably was difficult without a systematic method for identifying which parts of the solution were redundant. This observation motivates our automatic conciseness metric: instead of relying on local perceptions of verbosity, it uses an operational definition based on how much text can be removed while preserving the mathematical argument.

Metric correlation We further measure the Pearson correlations between the different dimensions. Overall, we find that computational ease is moderately correlated with elegance ($r = 0.51$), and cognitive simplicity is weakly correlated with insight ($r = 0.28$). Conciseness, likely due to its low agreement, is essentially orthogonal to the other dimensions.

Validation on computational ease and cognitive simplicity Using the 90 human judgments, we compare GPT-OSS-120B pairwise preferences against human preferences. GPT-OSS-120B agrees with human judgments at rates of 85.3% for *cognitive simplicity* and 82.1% for *computational ease*. These agreement rates are comparable to inter-annotator agreement. This provides evidence that the LLM judge applies these criteria consistently. One possible explanation for the higher scores is that the doubly annotated subset contains many pairs of similar solutions, where fine-grained distinctions may be more difficult for humans than for automated systems.

Table 5. Mean Rand Index between clustering on the full set of solutions and clustering on smaller subsets. We measure this for human-only solutions and for the combined human + LLM solution pool.

Method	Human	Human + LLM
Ours	87.8%	88.6%
Full	82.3%	79.4%
Fingerprint	79.1%	80.0%

Table 6. Agreement with GPT-5.4 reference judgments for different correctness-verification components.

Method	Agreement
Answer + Completeness	91.5%
Completeness	88.0%
Answer	73.0%

B.3. Clustering Consistency Validation

Our diversity and adaptivity metrics use an LLM-based clustering procedure to group solutions by their underlying mathematical approach. Because this procedure may depend on the set of solutions shown to the model and on the representation of each solution, we evaluate its consistency under subsampling and compare several input representations.

In the main experiments, the clustering model receives short summaries of all solutions. We compare this design with two alternatives:

- **Ours**: short summaries of all solutions.
- **Full**: full solution texts.
- **Fingerprint**: extracted keywords describing the backbone techniques and proof strategies used in each solution.

To measure consistency, we first cluster the full set of solutions for a problem. We sample up to five subsets per problem, choosing each subset size uniformly at random. For each subset, we restrict the full-set clustering to the same solutions and compute the Rand Index between the restricted full-set clustering and the subset clustering. We report the average Rand Index across subsets and problems. Results are shown in Table 5, separately for human-only solutions and for the combined human + LLM solution pool.

The short-summary representation is the most consistent across both settings. It achieves a mean Rand Index of 87.8% on human-only solutions and 88.6% on the combined human + LLM pool. The full-solution representation is less stable, especially in the larger combined pool, likely because long solutions introduce irrelevant details that distract from the core technique. This is particularly problematic for larger solution sets, where the context size may overwhelm the model. Conversely, the fingerprint representation appears too sparse: a small number of keywords often fails to capture the substantive relationship between two approaches, leading to inconsistent cluster assignments. These results motivate our choice of short summaries as a balance between informativeness and robustness.

B.4. Correctness Verifier Ablation

After validating quality judgments and clustering, we validate the correctness filter from §3.3 which consists of an answer checker plus a completeness verifier. The first checks the final answer and the second checks whether the reasoning justifies it. To test both components, we measure agreement with GPT-5.4, a strong standalone judge, using the full verification prompt in §F.3.3.

The results in Table 6 show that the combination of the two components is significantly better aligned than either component in isolation. These results are computed before manually resolving discrepancies between the verifier outputs and the LLM-based validation.

Table 7. Model ratings across the different PROOFRANK metrics (excluding Adaptivity) on solutions with correct answers (CA). We also show the differences in Elo with respect to the main results.

	Conciseness	**Computational Ease**	**Cognitive Simplicity**	**Diversity**
GPT-5.4	1219 (-160)	1390 (-225)	1211 (-119)	*1116 (-432)*
GEMINI-3.1-PRO	*415* (+143)	1002 (-44)	1086 (-91)	1222 (-112)
KIMI-K2.5-THINK	1418 (-117)	1181 (-70)	*1067* (-43)	**1275 (+131)**
GLM-5	871 (-66)	*926* (+61)	1204 (+9)	1193 (-19)
DEEPSEEK-V3.2	1211 (-39)	1137 (+85)	1282 (+13)	1239 (+112)
STEP-3.5-FLASH	1405 (0)	1203 (-17)	1171 (+10)	1228 (-13)
GPT-OSS-120B	1306 (-146)	1064 (-55)	1070 (-46)	1237 (+158)
GROK-4.1-FAST	**1604** (+134)	1231 (+219)	1300 (+137)	1119 (+10)
GEMINI-3-FLASH	1366 (+185)	1337 (+24)	1283 (+26)	1153 (+49)
QWEN3.5-397B-A17B	1185 (+67)	**1527** (+19)	**1326** (+105)	1218 (+121)

In the 17 samples where the verifier and GPT-5.4 disagreed, we found through manual inspection that our verifier was correct in 4 cases, while GPT-5.4 was correct in the remaining 13. These errors are classified into 10 false positives and 3 false negatives, which fall into the following categories:

- In **10 solutions**, the completeness verifier was too permissive and accepted solutions that omitted an important reasoning step. In most of these cases, however, the solution followed a technique that could plausibly lead to a correct proof if completed.
- In **2 solutions**, the completeness verifier was too strict and rejected arguments that reused an analogous reasoning pattern multiple times. Although this is a valid proof strategy, the verifier expected a more explicit statement of the analogy.
- In **1 solution**, a geometry problem from IMO-ANSWERBENCH contained a confusing rephrasing. The solution correctly identified that one point had not been translated correctly from the original statement. However, because recovering the intended statement required an additional assumption, the completeness verifier rejected the solution.

Overall, these errors affect only a small fraction of the evaluated samples. Moreover, many false positives correspond to incomplete but directionally correct solution strategies. We therefore expect the verifier errors to have limited impact on the main results.

C. Additional Results

In this section, we provide additional experimental results that complement the evaluations presented in §4.

C.1. PROOFRANK on Correct-Answer Solutions

We repeat the experiments presented in §4.4, but now restrict the evaluation to solutions verified to have a correct final answer, rather than removing the correctness dependence entirely. As shown in Table 7, the rating differences under this restriction follow a trend similar to those without any correctness filtering. The differences are less pronounced, but retain the same direction. The only exception is STEP-3.5-FLASH, which remains at approximately the same ratings as in the main results (§4.1). This suggests that its fully incorrect solutions are likely substantially different in style, being less verbose and overall simpler, which leads to a major ranking shift when they are included.

C.2. Impact of Rephrasing Validity on Conciseness

As we explain in §3.2, we use a strong reasoning LLM to shorten solutions while preserving their completeness and correctness, which allows us to compute our conciseness metric. However, this introduces the risk that the rephrasing process may add or omit important details. To assess this, we use an LLM to automatically determine whether the original and rephrased solutions are fully equivalent, disregarding minor stylistic differences.

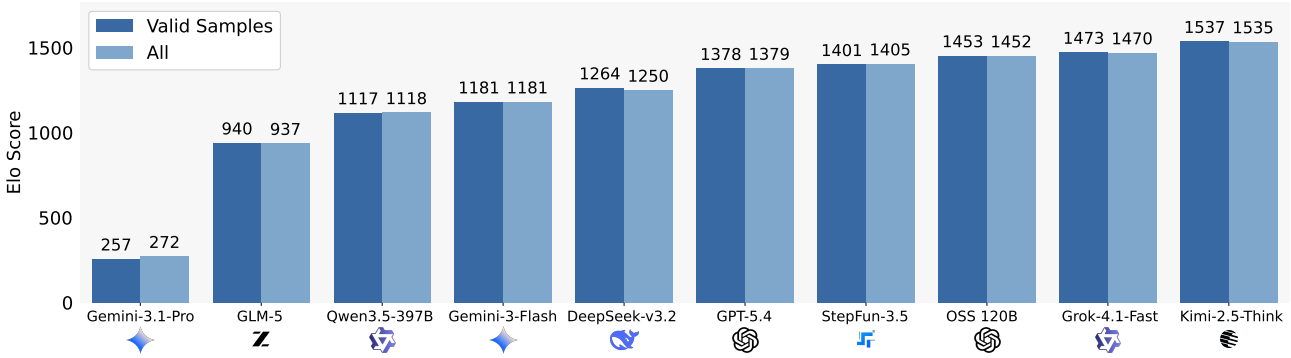

Figure 5. Elo changes when only considering valid rephrasings for the conciseness metric.

Table 8. Main model ratings across the different PROOFRANK metrics, measured on correct solutions only, alongside their accuracy rate. We include 95% confidence intervals for all metrics in the superscripts.

	Conciseness	Computational Ease	Cognitive Simplicity	Diversity	Adaptivity	Accuracy
GPT-5.4	1379^{+40}_{-41}	$\mathbf{1615^{+46}_{-48}}$	$\mathbf{1330^{+34}_{-32}}$	1261^{+38}_{-40}	$\mathbf{1558^{+14}_{-15}}$	$\mathbf{86.4^{\pm 3.4}}\%$
GEMINI-3.1-PRO	$\mathit{272^{+60}_{-75}}$	1046^{+45}_{-40}	1177^{+34}_{-30}	$\mathbf{1310^{+40}_{-42}}$	1334^{+12}_{-11}	$60.7^{\pm 4.9}\%$
KIMI-K2.5-THINK	$\mathbf{1535^{+47}_{-43}}$	1251^{+50}_{-46}	$\mathit{1110^{+34}_{-35}}$	1272^{+47}_{-47}	1144^{+11}_{-11}	$53.6^{\pm 5.0}\%$
GLM-5	937^{+46}_{-49}	$\mathit{865^{+50}_{-52}}$	1195^{+33}_{-34}	1132^{+39}_{-37}	1212^{+12}_{-11}	$49.1^{\pm 5.0}\%$
DEEPSEEK-V3.2	1250^{+51}_{-50}	1052^{+56}_{-59}	1269^{+39}_{-38}	1285^{+54}_{-53}	1127^{+11}_{-11}	$47.9^{\pm 5.0}\%$
STEP-3.5-FLASH	1405^{+49}_{-46}	1220^{+53}_{-54}	1161^{+37}_{-35}	1170^{+38}_{-40}	1241^{+12}_{-11}	$41.1^{\pm 4.9}\%$
GPT-OSS-120B	1452^{+51}_{-42}	1119^{+47}_{-52}	1116^{+34}_{-34}	1205^{+40}_{-39}	$\mathit{1079^{+11}_{-11}}$	$34.3^{\pm 4.8}\%$
GROK-4.1-FAST	1470^{+47}_{-49}	1012^{+51}_{-51}	1163^{+36}_{-41}	$\mathit{1103^{+41}_{-43}}$	1109^{+11}_{-11}	$31.4^{\pm 4.6}\%$
GEMINI-3-FLASH	1181^{+48}_{-46}	1313^{+51}_{-52}	1257^{+35}_{-35}	1110^{+41}_{-44}	1104^{+11}_{-12}	$29.8^{\pm 4.6}\%$
QWEN3.5-397B-A17B	1118^{+51}_{-53}	1508^{+61}_{-55}	1221^{+35}_{-37}	1153^{+39}_{-41}	1091^{+11}_{-11}	$\mathit{29.3^{\pm 4.6}}\%$

We find that 94.1% of the rephrasings are fully valid, with the remaining 5.9% potentially affecting the reliability of the conciseness metric. We examine the resulting Elo changes when only considering valid rephrasings in Figure 5, which shows that the Elo differences are minimal for all models, with no changes larger than 10 to 15 points. This change is much smaller than the sampling error (§C.3), suggesting that our conclusions persist under both settings.

C.3. Confidence Intervals for Main Results

To assess the statistical significance of our main results, we compute 95% confidence intervals for all Elo scores and accuracy rates in Table 8. We do so by performing 1000-fold bootstrapping on the pairwise judgments for the quality metrics and by assuming a binomial distribution for the accuracy rates.

D. Methodological Challenges and Failed Attempts

Throughout the development of our work, we explored a variety of methodologies that are commonly used in the LLM evaluation literature. However, we found that many of these approaches were ineffective for evaluating aspects that are either too subjective or too nuanced for LLMs to assess consistently and accurately. In this section, we describe several of these attempted approaches and explain why they failed, both to contextualize our final design choices and to provide guidance for future work in this area.

D.1. Ordinal Ranking of Cognitive Simplicity and Computational Ease

Before adopting our pairwise comparison-based evaluation for cognitive simplicity and computational ease, we experimented with asking LLMs to assign scores to solutions on an ordinal scale. To do so, we designed a rubric for each proof aspect. For example, our cognitive simplicity rubric identified the most challenging proof concept in a solution and ranked it by the mathematical background required: for example, middle-school, IMO-level, or advanced research mathematics.

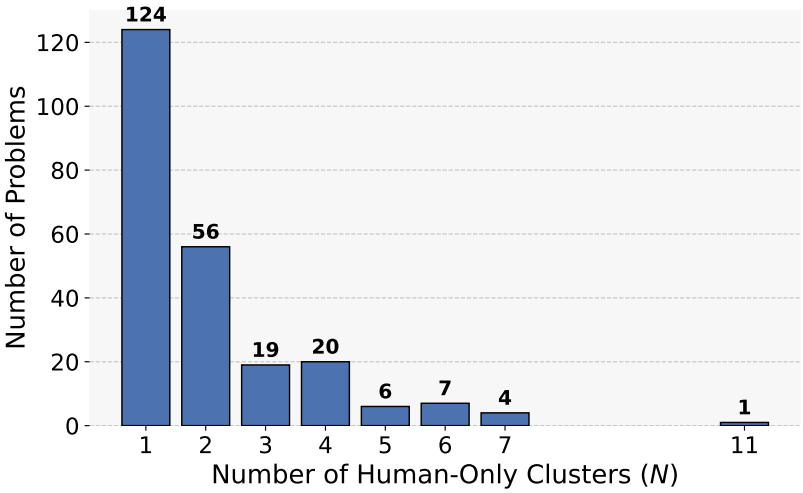

Figure 6. Distribution of the number of solutions per problem in our dataset.

We found such rubrics difficult to use in our setting for two main reasons:

- LLMs are not necessarily familiar with the background assumptions required by the rubric, and can therefore struggle to apply it consistently.
- Designing a comprehensive evaluation rubric is inherently difficult, as many nuances and edge cases cannot be captured by a fixed set of rules, which increases noise and introduces potential bias.

We encountered similar challenges in the human evaluation described in §B.2, where we initially attempted to rate each aspect on a scale from 0 to 5. Not only was it difficult to design an appropriate rubric for such ratings, but the subjective nature of the task also introduced substantial annotation noise, making it difficult to draw meaningful conclusions.

D.2. Evaluation on Proof-based Problems

In an initial stage of our work, we considered constructing PROOFRANK from proof-based problems rather than restricting ourselves to final-answer problems. However, validating the correctness of proof-based solutions in a scalable manner is extremely challenging. Although (Dekoninck et al., 2025) find that frontier LLMs perform well as correctness judges, these models are often expensive to use, making it financially infeasible to scale such evaluation to a benchmark of our size.

We also attempted to use smaller models as verifiers, such as OPC-R1-8B, GROK-4.1-FAST, and GPT-OSS-120B, which are either open-weight or more affordable. However, we found that these models were either insufficiently accurate or significantly biased in favor of certain models over others. Such biases would substantially skew our results, so we decided against using LLMs as correctness judges.

Nevertheless, we still wanted to account for proof correctness in our evaluation, since final-answer solutions that do not present a complete proof are of limited value to the mathematical community. We therefore adopted the two-step correctness verification framework described in §3.3.

E. Additional Dataset Information

In this section, we describe the procedure used to construct the problem set for PROOFRANK. For competitions imported from MathArena, we exclude problems that are likely to be too straightforward to elicit meaningfully distinguishable model solutions. As a heuristic, we remove the first five problems from each AIME competition, treating AIME I and AIME II separately, and the first three problems from each HMMT category, with problems 1 to 10 corresponding to Algebra, 11 to 20 to Geometry, and so on. For IMO-ANSWERBENCH and the Apex/Apex Shortlist problems, we retain all problems except for four drawn from AIME 2025 or HMMT 2025, which are already included through the corresponding competition sources.

Table 9. A list of sources for the problems in PROOFRANK.

Competition	Description	Problems Included	Source
MathArena			
AIME 2025	Answer-based competition, serving as a qualifier for the USAMO	6-15,21-30	Public
HMMT February 2025	Answer-based competition hosted by Harvard and MIT	4-10,14-20,24-30	Public
AIME 2026	Answer-based competition, serving as a qualifier for the USAMO	6-15,21-30	Public
HMMT February 2026	Answer-based competition hosted by Harvard and MIT	4-10,14-20,24-33	Public
Apex 2025	Selected final-answer problems, which LLMs struggle to solve	All (12)	Public
Apex Shortlist 2025	Selected final-answer problems, which LLMs find more challenging	All (45)	Public
IMO-AnswerBench			
Various	High-school-level competitions, from which problems were rephrased.	241 problems	Public

For each problem, we collect human-written solution attempts from public Art of Problem Solving (AoPS) posts or the official competition sources. We discard problems for which no public solution could be found. This filtering removed approximately 100 problems from the original IMO-ANSWERBENCH dataset of 400 problems. In addition, some IMO-ANSWERBENCH problems are substantially rephrased relative to their source problems. We remove approximately 60 such problems when the original solution strategy does not trivially transfer to the rephrased version. After these filtering steps, 241 IMO-ANSWERBENCH problems remain. The full dataset breakdown is reported in Table 9.

For adaptivity, we use these public solutions to sample up to 4 candidate techniques per problem, as described in §3. This limits inference costs and prevents problems with unusually many available solution strategies from dominating the results. As shown in Figure 6, the number of extracted techniques varies substantially across problems. The cap also mitigates the overrepresentation of geometry problems, which account for a large fraction of high-technique-count problems because many synthetic and algebraic solutions are possible.

F. Prompts and Instructions

In this section, we provide the specific prompts used throughout our experiments for reproducibility.

F.1. Human Validation Instructions

Below we include the instructions provided to the human annotators for pairwise comparisons of the solutions on the different metrics.

Verbosity

```
Decide which solution is more verbose: 0 = left, 1 = right. Judge verbosity by how much text is used to express the same content (
extra explanations, repeated statements, long-winded phrasing, unnecessary intermediate steps), not by mathematical difficulty. If
they feel equally verbose, pick the one that spends more words on redundant detail.
```

Elegance

```
Decide which solution is more elegant: 0 = left, 1 = right. High elegance feels clean and unified: it uses a clever trick, symmetry,
an invariant, or a principled shortcut that avoids heavy calculation and produces an "aha" moment. Standard elegance uses the expected
 method efficiently but without surprise. Low elegance is brute force: long algebra/coordinate bashing when a cleaner approach exists,
 large/manual casework, or repetitive computations that obscure the main idea. If both are similarly elegant, choose the one with
fewer moving parts and a clearer unifying idea.
```

Insightfulness

```
Decide which solution is more insightful: 0 = left, 1 = right. An insightful solution exposes a deeper structure, gives a clear "why
this works" viewpoint, or makes a meaningful connection between ideas/areas (e.g., reframing number theory via combinatorics/
invariants/geometry), and leaves the reader with a reusable lesson. A less insightful solution is mainly routine manipulation or
casework, follows a standard template, or applies a "black-box' theorem/formula with little explanation of why it is the right fit. If
 both feel equally insightful or there isn't enough information to tell, pick the one that better explains the core idea rather than
```

```
the one with more steps.
```

Cognitive Simplicity

```
Decide which solution is more conceptually complex (more ingenuity/non-obvious core idea, more specialized/advanced tools, or harder
integration of multiple ideas). Ignore notation density, computation/bookkeeping time, and step count unless it reflects genuinely
different conceptual ingredients. Output exactly one number and nothing else: 0 = Solution 1 is more complex, 0.5 = Tie/unclear from
the text, 1 = Solution 2 is more complex.
```

Computational Ease

```
Decide which one takes more time to follow line-by-line due to:
 - heavier notation/symbol tracking (many variables/indices, dense formulas, switching conventions, overloaded symbols),
 - unclear or inconsistent definitions/references (symbols used before defined, ambiguous bounds/domains/constraints),
 - more explicit computation/bookkeeping (long chains of routine steps, expansions, substitutions, sign/exponent/index tracking).

Ignore conceptual cleverness, focus only on time-to-parse/track what is written.
 - 0 = Solution 1 imposes more time/load
 - 0.5 = Tie / cannot confidently rank from the text
 - 1 = Solution 2 imposes more time/load
```

F.2. Solution Generation Prompts

We use the following prompts to generate solutions for our evaluation. This includes our standard prompt that follows (Dekoninck et al., 2025), as well as variations for specific experiments, such as technique-constrained generation and 2 prompts for emphasizing conciseness.

Standard Generation Prompt

```
Your task is to write a proof solution to the following problem, focusing on accuracy, thoroughness, and clarity. When you write your
proof, follow these guidelines:

- You are creating a proof, not a proof outline. Each step should be carefully explained and documented. If not properly explained,
the judge will assume that you cannot explain it, and therefore decrease your grade.
- You can use general theorems and lemmas, but only if they are well-known. As a rule of thumb: if the result has a name and is famous
 enough to have a Wikipedia page or something similar to describe it, it is allowed. Any result from papers that would not be taught
in high-school or low-level bachelor courses in mathematics should not be used.
- Do not skip computation steps in your proof. Clearly explain what transformations were done and why they are allowed in each step of
 a calculation.
- Your proof should be self-contained.
- If you are not sure about a specific step, or do not know how to prove an intermediate result, clearly state this. It is much
preferable to indicate your uncertainty rather than making incorrect statements or claims.
- Put your final answer within \\boxed{{}}.

{problem}
```

Techinque-constrained Generation Prompt

```
Your task is to write a proof solution to the following problem. You will be required to solve the problem using a specific solving
technique. When you write your proof, follow these guidelines:

- You are creating a proof, not a proof outline. Each step should be carefully explained and documented. If not properly explained,
the judge will assume that you cannot explain it, and therefore decrease your grade.
- You can use general theorems and lemmas, but only if they are well-known. As a rule of thumb: if the result has a name and is famous
 enough to have a Wikipedia page or something similar to describe it, it is allowed. Any result from papers that would not be taught
in high-school or low-level bachelor courses in mathematics should not be used.
- Do not skip computation steps in your proof. Clearly explain what transformations were done and why they are allowed in each step of
 a calculation.
- Your proof should be self-contained.
- Your proof should make use of the specified technique. You will not receive any credit if you fail to use the specified technique.
- If you are not sure about a specific step, or do not know how to prove an intermediate result, clearly state this. It is much
preferable to indicate your uncertainty rather than making incorrect statements or claims.
- Put your final answer within \\boxed{{}}.

{problem}

Solve the problem using the following method: {technique}
```

Generation Prompt for Conciseness

```
Your task is to write a proof solution to the following problem, focusing on accuracy, thoroughness, and clarity. You should make sure
 that your proof contains as little extra comments as possible, and is as concise as possible so long as the proof is correct. When
you write your proof, follow these guidelines:

- You are creating a proof, not a proof outline. Each step should be carefully explained and documented. If not properly explained,
the judge will assume that you cannot explain it, and therefore decrease your grade.
- You can use general theorems and lemmas, but only if they are well-known. As a rule of thumb: if the result has a name and is famous
 enough to have a Wikipedia page or something similar to describe it, it is allowed. Any result from papers that would not be taught
in high-school or low-level bachelor courses in mathematics should not be used.
- Do not skip computation steps in your proof. Clearly explain what transformations were done and why they are allowed in each step of
 a calculation.
- Your proof should be self-contained.
- Make sure the proof is as concise as possible while still being correct. Avoid additional explanations or comments that are not
strictly necessary for the correctness of the proof.
- Put your final answer within \\boxed{{}}.

{problem}
```

Generation Prompt for Stronger Conciseness

```
Your task is to write a proof solution to the following problem, focusing on conciseness. You should make sure that your proof
contains as little extra comments as possible, and is as concise as possible so long as the proof is correct. When you write your
proof, follow these guidelines:

  - You are creating a proof, not a proof outline. Each step should be carefully explained and documented.
  - Make sure the proof is as concise as possible while still being correct. Avoid additional explanations or comments that are not
  strictly necessary for the correctness of the proof.

  {problem}

{problem}
```

F.3. Correctness Verification Prompts

F.3.1. Solution Verification Prompt

We use the following prompt for running GPT-5.4 to determine the overall correctness of a mathematical solution and construct .

Solution Verification Prompt

```
You are judging the correctness of an LLM-generated proof for a math problem.

### Input:

Your input will consist of the following components:
- **Problem Statement**: A mathematical problem that the proof is attempting to solve.
- **Proof Solution**: The proof that you need to evaluate. This proof may contain errors, omissions, or unclear steps. The proof was
generated by another language model, which was given the following instructions:
<model_prompt>
- You are creating a proof, not a proof outline. Each step should be carefully explained and documented. If not properly explained,
the judge will assume that you cannot explain it, and therefore decrease your grade.
- You can use general theorems and lemmas, but only if they are well-known. As a rule of thumb: if the result has a name and is famous
 enough to have a Wikipedia page or something similar to describe it, it is allowed. Any result from papers that would not be taught
in high-school or low-level bachelor courses in mathematics should not be used. Any use of such results will immediately give you a
zero grade.
- Do not skip computation steps in your proof. Clearly explain what transformations were done and why they are allowed in each step of
 a calculation.
- You should use correct LaTeX notation to write equations and mathematical symbols. You should encompass these equations in
appropriate symbols ("\\(" and "\\)" for inline math, "\\[" and "\\]" for block math) to enhance the clarity of your proof. Do not use
 any unicode characters.
- Your proof should be self-contained.
- If you are not sure about a specific step, or do not know how to prove an intermediate result, clearly state this. It is much
preferable to indicate your uncertainty rather than making incorrect statements or claims.
</model_prompt>

### How the solution should be graded:
A solution should be considered correct even if it would earn 5+/7 points in a standard grading format. Examples of small penalties
worth 1 point are if the solution:
- Makes a small computational mistake that can be easily fixed
```

```
- Misses an edge case which can be easily proven/disproven
- Skips over a step that follows without much reasoning or manual work
Depending on the severity and the context, you may also not penalise a given error. On the other hand, a solution should be marked as
incorrect if:
- It marks a step as trivial, if it is not immediately obvious with little reasoning why this would be the case.
- It omits algebra-heavy computational steps, regardless of whether or not it has outlined the methodology. Skipping shorter
computations should be permitted.
- Generalizes over a pattern without rigorously describing the pattern, or without proving any relevant properties.
- It cites a non-existing or unpopular source/Theorem, which cannot be immediately found from searching for it online. Thus, any
theorems that can be immediately found and have a Wikipedia article are allowed.

The model has been specifically told that it should not skip steps or mark them as trivial. Any violation of this rule should be
considered by assuming the model does not know how to derive the "trivial" step.

### Scoring instructions

If you believe the proof is correct, end your analysis with \\boxed{{correct}}. If you believe the proof is incorrect, end your
analysis with \\boxed{{incorrect}}.

### Problem Statement:
{problem}

### Model Solution:
{solution}
```

F.3.2. Answer Verification Prompt

Following (Luong et al., 2025), we use the following prompt to verify the correctness of the final answer provided in a model solution.

Answer Verification Prompt

```
# System Role: Deterministic Mathematical Autograder
You are a precise, automated grading system. Your sole function is to determine if the final answer provided in the Model Solution is
mathematically equivalent to the Golden Answer. You must NOT grade the reasoning or steps, only the final result.

# 1. Grading Guidelines (Equivalence Rules)

Equivalence is mandatory for a correct grade. You must rigorously verify if the answers represent the exact same mathematical value or
 expression, even if the format differs.

* **Algebraic Equivalence:** e.g., 'n(n+1)/2' is equivalent to 'n^2/2 + n/2'. You must verify the algebra.

* **Numerical Equivalence:** e.g., '1/2' is equivalent to '0.5'; 'sqrt(2)/2' is equivalent to '1/sqrt(2)'.

* **Set/List Equivalence:** Unless specified as an ordered tuple/vector, the order of elements does not matter (e.g., {{1, 2}} is
equivalent to {{2, 1}}).

* **Partial Credit:** No partial credit is allowed. If the answer is incomplete or partially incorrect, it is incorrect.

* **No Answers:** If no clear, unambiguous final answer can be extracted, the solution must be graded as incorrect.

# 3. Output Protocol (Strict Compliance Required)

You must execute the task using a two-part structure. Failure to follow this structure will result in task failure.

**Part 1: Analysis (Chain-of-Thought)**

You MUST perform your analysis within <thinking></thinking> tags. Make your thinking concise. This section details your reasoning
process and must follow these steps sequentially:

1. **Golden Answer:** State the Golden Answer.

2. **Extracted Model Answer:** State the extracted answer based on the Extraction Protocol. If none found, state "No clear final
answer found."

3. **Equivalence Analysis:** Compare the two answers using the Grading Guidelines. Detail the steps taken to verify mathematical
equivalence (e.g., simplification, algebraic manipulation). You must actively try to prove they are the same before concluding they
are different.

4. **Conclusion:** State the final determination ("Correct" or "Incorrect").

**Part 2: Final Grade**

Immediately following the closing </thinking> tag, output **ONLY** the final grade.
```

```
* If Correct: \\boxed{{Correct}}
* If Incorrect: \\boxed{{Incorrect}}

**CRITICAL CONSTRAINT: Do not add any text, explanations, or formatting outside the <thinking> tags or the final \\boxed{{}} output.**

-------
Output exmaple:
<thinking>
1. **Golden Answer:** (-\infty, -4) \cup (4, \infty)
2. **Extracted Model Answer:** ∅(the empty set)
3. **Equivalence Analysis:**
The Golden Answer is a non-empty set of real numbers. The Model Answer is the empty set. These two sets are not equivalent. The empty
set contains no elements, while the Golden Answer contains an infinite number of elements.

4. **Conclusion:** Incorrect
</thinking>
\\boxed{{Incorrect}}

-------
# 4. Input Data
Here is the problem, model solution, and golden answer to grade:

### Problem Statement:
{problem}

### Model Solution:
{solution}

### Golden Answer:
{gold_answer}
```

F.3.3. Solution Verification Prompt

We use the following prompt for running GPT-5.4 to determine the overall correctness of a mathematical solution and construct .

Solution Verification Prompt

```
You are judging the completeness of an LLM-generated proof for a math problem.

### Input:

Your input will consist of the following components:
- **Problem Statement**: A mathematical problem that the proof is attempting to solve.
- **Proof Solution**: The proof that you need to evaluate. This proof may contain errors, omissions, or unclear steps. The proof was
generated by another language model, which was given the following instructions:
<model_prompt>
- You are creating a proof, not a proof outline. Each step should be carefully explained and documented. If not properly explained,
the judge will assume that you cannot explain it, and therefore decrease your grade.
- You can use general theorems and lemmas, but only if they are well-known. As a rule of thumb: if the result has a name and is famous
 enough to have a Wikipedia page or something similar to describe it, it is allowed. Any result from papers that would not be taught
in high-school or low-level bachelor courses in mathematics should not be used. Any use of such results will immediately give you a
zero grade.
- Do not skip computation steps in your proof. Clearly explain what transformations were done and why they are allowed in each step of
 a calculation.
- You should use correct LaTeX notation to write equations and mathematical symbols. You should encompass these equations in
appropriate symbols ("\\(" and "\\)" for inline math, "\\[" and "\\]" for block math) to enhance the clarity of your proof. Do not use
 any unicode characters.
- Your proof should be self-contained.
- If you are not sure about a specific step, or do not know how to prove an intermediate result, clearly state this. It is much
preferable to indicate your uncertainty rather than making incorrect statements or claims.
</model_prompt>

### How the solution should be graded:
A solution should be considered complete if it has described every step of the proof with clear logical reasoning. Do NOT verify the
overall correctness of the proof, but only check if sufficient details are presented to make the proof complete.

#### Small mistakes that do not affect the overall completeness of the proof:
- Makes a small computational mistake that can be easily fixed
- Misses an edge case which can be easily proven/disproven
- Skips over a step that follows without much reasoning or manual work

#### The following mistakes should be considered as making the proof incomplete:
- It marks a step as trivial, if it is not immediately obvious with little reasoning why this would be the case.
- It omits algebra-heavy computational steps, regardless of whether or not it has outlined the methodology. Skipping shorter
computations should be permitted.
```

```
- Generalizes over a pattern without rigorously describing the pattern, or without proving any relevant properties.
- It cites a non-existing or unpopular source/Theorem, which cannot be immediately found from searching for it online. Thus, any
theorems that can be immediately found and have a Wikipedia article are allowed.

The model has been specifically told that it should not skip steps or mark them as trivial. Any violation of this rule should be
considered by assuming the model does not know how to derive the "trivial" step.

### Scoring instructions

If you believe the proof is complete, end your analysis with \\boxed{{complete}}. If you believe the proof is incomplete, end your
analysis with \\boxed{{incomplete}}.

### Problem Statement:
{problem}

### Model Solution:
{solution}
```

F.4. Clustering Prompts

Solution Summary Generation We use the following prompt to cluster different solutions for easier parsing by the diversity clustering LLM. We find that this significantly improves the consistency of the clustering, as it provides a more concise and structured overview of the solutions, allowing the model to better identify the core techniques used in each solution, without getting lost in the details of the full solution text.

Solution Summary Generation Prompt

```
You are an expert mathematical annotator tasked with extracting the core method fingerprint of a provided solution.

# Main task

Identify the central mathematical idea that makes this particular solution work.

Focus on the method actually present in the provided solution text.
Do not solve the problem yourself.
Do not repair missing steps.
Do not add steps that are not explicitly supported by the solution.

# Analysis

Analyze the solution at three levels:

1. **Family / viewpoint**
The broad style, such as algebraic, geometric, combinatorial, constructive, extremal, invariant-based.

2. **Backbone**
The main structural move that sets up the solution, such as a reflection construction, polynomial-root encoding, greedy maintenance
strategy, block decomposition, reduction to a theorem, or symmetry-based transformation.

3. **Closing engine**
The theorem, lemma, or decisive move that turns the backbone into the conclusion.

The **backbone** is the most important field.
The **closing engine** should be recorded when it is a real mathematical ingredient, not just routine simplification.

# Requirements

1. Identify what makes the solution work conceptually, not how to carry out the computations.
2. Do not solve the problem on your own.
3. Do not include chain-of-thought, hidden reasoning, or speculative reconstruction.
4. Do not fill in missing steps.
5. Work only with what is explicitly present or very strongly signaled in the solution text.
6. If the solution is incomplete but its intended method is still recognizable, describe that intended method.
7. If the text is too vague to identify a genuine nontrivial method, set `noCoreIdea` to `true` and use `null` where appropriate.
8. Avoid generating equations or calculations yourself. Short verbatim evidence quotes may contain formulas if they appear in the
provided solution.
9. Write `coreIdea` and `supportingIdeas` in an imperative, hint-like style.
10. Keep labels method-specific. Avoid broad labels such as "use algebra," "use symmetry," or "use geometry" unless no more specific
description is supported.

# Output format

Your response must include:

- **noCoreIdea**: whether the provided solution lacks an identifiable nontrivial core idea
```

```
- **solutionSummary**: a concise summary of the solution's method and structure, written descriptively
- **coreIdea**: one short imperative sentence giving the main hint
- **supportingIdeas**: zero to three short imperative phrases for secondary ideas
- **family**: the broad mathematical viewpoint
- **backbone**: the main structural move that defines the method
- **closingEngine**: the decisive theorem, lemma, or final mechanism, or `null` if none is identifiable
- **evidenceQuotes**: one to three short quotes or pinpointed references from the solution supporting your identification
- **confidence**: a number between 0 and 1

# Style guidelines

- `solutionSummary` should be descriptive, compact, and reconstruction-friendly.
- `coreIdea` should be short, concrete, and method-level.
- `supportingIdeas` should name supporting moves, not computations.
- Do not mention steps that are absent from the solution.
- Do not overstate confidence when the proof is sketchy or incomplete.

# Schema

```json
{{
"noCoreIdea": false,
"solutionSummary": "",
"coreIdea": "",
"supportingIdeas": [],
"family": "",
"backbone": "",
"closingEngine": null,
"evidenceQuotes": [],
"confidence": 0.0
}}
```
Notes

- Use `noCoreIdea = true` only when the solution text does not support a recognizable central method.
- If the proof is short but clearly method-driven, do **not** mark it as lacking a core idea.
- If the proof names a theorem but does not actually use it, do not treat that theorem as the backbone or closing engine.
- If two descriptions would differ only in wording, prefer the more concrete mathematical description.

Input

Problem
{problem}

Solution
{solution}
```

Solution Summary Clustering We use the following prompt to cluster different solutions based on their solving techniques.

Solution Clustering Prompt

```
Your task is to analyze multiple candidate proofs of the same problem and cluster them by their core mathematical method.

# Main Goal

Given one problem and N proof summaries, determine which proofs are essentially the same method and which are meaningfully different
methods.

Your job is not to grade correctness or completeness, but to detect diversity of methods in a way that is stable and consistent.

# Pairwise consistency rule

For any pair of proofs, the same-cluster / different-cluster decision should be the same whether those two proofs are shown alone or
inside a larger batch, regardless of the presence of other proofs.

# Method analysis framework

Analyze each prof at three levels:

1. **Family / viewpoint**
The broad style, such as algebraic, geometric, combinatorial, constructive, extremal, invariant-based.

2. **Backbone**
The main structural move that sets up the proof, i.e. a geometric construction, polynomial-root encoding, decomposition, extremal
choice, or reduction to a known theorem.
```

```
3. **Closing engine**
The theorem, lemma, or decisive move that links the backbone and the conclusion.

## Clustering rule
Cluster primarily by the **backbone**.

Use the **closing engine** to split proofs only when it is an essential, non-routine reason the proof works.

Do **not** cluster by broad family alone.

# Pairwise clustering criteria

## Put two proofs in the same cluster if:
- they share the same backbone, and
- they reach the result for the same essential mathematical reason,
even if they differ in wording, notation, order, level of detail, or minor helper tools.

## Put two proofs in different clusters only if you can clearly state:
- "One proof hinges on X, while the other hinges on Y,"
or
- "One proof translates the problem into X, while the other instead uses Y,"

If you cannot articulate a clear method-level difference, prefer merging.

# Differentiating the proof scaffold vs engine

If two proofs share the same setup or initial construction, do not automatically merge them. Instead, check whether they differ in the
 following manner:
- Same scaffold + same essential reason for finishing => same cluster
- Same scaffold + different essential non-routine finishing engine => different clusters
- Same scaffold + different routine algebra => same cluster

# Determining equivalent methods

Treat two proofs as the same method when they use mathematically equivalent realizations of the same backbone, even if they:
- introduce different auxiliary objects
- phrase the same construction differently
- derive the same key relation through different local manipulations
- present the same symmetry or transform through different language

If the mathematical role is the same, prefer merging.

# What does not make a new cluster

Do not split proofs just because of:
- different notation or variable names
- different exposition style
- different order of steps
- one proof being more polished or detailed
- naming a theorem versus using it implicitly
- minor local casework inside the same global plan
- different algebraic packaging of the same derived relation
- a cosmetic lemma that merely restates the key step
- a different auxilary object playing the same structural role

# What makes a new cluster

Split proofs when they have a genuinely different:
- strategic paradigm
- representation of the problem
- backbone construction or reduction
- invariant, extremal quantity, or inductive mechanism
- decisive theorem or lemma, if that theorem is the real engine
- constructive mechanism
- reason the proof closes

# Cluster naming rule

Labels such as "algebraic," "geometric," "reflection," "greedy," "Vieta," "casework," or "symmetry" are too broad by themselves.

A cluster name should identify the actual backbone and, when essential, the closing engine.

# Incorrect or incomplete proofs

- If a proof is incorrect but clearly intends a recognizable method, cluster it by intended method.
- If it is too vague to identify a real method, place it in an `Unclear / Non-proof` cluster.
- Do not invent missing key steps.
- Do not assume a standard argument unless the text supports it.
```

```
# Pipeline

## Step 1: Build a method fingerprint for each proof
For each `Proof i`, identify:
- primary approach
- secondary techniques
- key pivot step
- evidence quotes

The fingerprint must be based only on what appears in the proof.

## Step 2: Make pairwise equivalence judgments
For each pair, ask:
"Would a knowledgeable mathematician describe these as the same proof idea?"

This judgment must depend only on the two proofs, not on the rest of the batch.

## Step 3: Form clusters
Create clusters from those pairwise judgments.

Each cluster must include:
- a concise cluster name
- the defining approach
- member proofs
- a short list of distinct features

## Step 4: Consistency check
Before finalizing:
- merge clusters that differ only in phrasing
- split clusters that hide clearly different essential engines
- check whether any near-duplicate proofs were accidentally separated
- check whether any proofs were merged only because they share a broad family label

## Output format

Return exactly the following JSON shape, with no extra fields and no commentary outside the JSON.

```json
{{
"N": 0,
"K": 0,
"diversity_score_D": 0.0,
"clusters": [
   {{
   "cluster_id": "C1",
   "cluster_name": "",
   "defining_approach": "",
   "defining_features": ["", ""],
   "members": [1]
   }}
],
"proof_fingerprints": [
   {{
   "proof_id": 1,
   "primary_approach": "",
   "secondary_techniques": ["", ""],
   "key_pivot_step": "",
   "evidence_quotes": ["", ""]
   }}
],
"warnings": ["", ""]
}}
```

Input

Problem

{problem}

Solutions

{solution_summary}
```

F.5. Topic Classification Prompt

We use the following prompt to classify each problem into one of four mathematical topics: Algebra, Geometry, Number Theory, and Combinatorics.

Topic Classification Prompt

```
Your task is to classify the following math problem into one of several topics based on its content and the mathematical concepts it
involves. The topics to choose from are:
- Algebra
- Geometry
- Number Theory
- Combinatorics
- Calculus

Output the topic that best fits the problem in \\boxed{{}}, i.e. \\boxed{{Algebra}}, at the end of your resposne.

Here are the problem statement and solution:

Problem:

{problem}

Solution:

{solution}
```

F.6. Solution Shortening Prompt

We use the following prompt to shorten proofs generated by LLMs to estimate the textual conciseness of the original solution.

Solution Shortening Prompt

```
Your task is to rephrase the following solution to a mathematical problem to make it less verbose without omitting any important
details. Ensure that the rephrased solution remains clear and accurate while being more concise. Follow these guidelines:
- Retain all critical steps and explanations necessary for understanding the solution.
- Do not introduce any new information or change the meaning of the original solution.
- The new solution must be equivalent to the original in terms of correctness and completeness.
- Do not omit any computations that were critical to the solution.


Original Problem:
{problem}

Original Solution:
{solution}
```

F.7. Shortening Verification Prompt

We use the following prompt to verify whether the shortened proofs are equivalent to the original.

Shortening Verification Prompt

```
Your task is to rephrase the following solution to a mathematical problem to make it less verbose without omitting any important
details. Ensure that the rephrased solution remains clear and accurate while being more concise. Follow these guidelines:
- Retain all critical steps and explanations necessary for understanding the solution.
- Do not introduce any new information or change the meaning of the original solution.
- The new solution must be equivalent to the original in terms of correctness and completeness.
- Do not omit any computations that were critical to the solution.


Original Problem:
{problem}

Original Solution:
{solution}
```

F.8. Pairwise Comparison Prompts

Pairwise Cognitive Simplicity Comparison Prompt We use the following prompt to compare the cognitive simplicity of different solutions based on their solving techniques.

Pairwise Cognitive Simplicity Comparison Prompt

```
You are an impartial LLM-as-a-judge. Your task is to compare Solution 1 vs Solution 2 and decide which one exhibits MORE **ideological
 complexity** to a technically competent reader, specifically due to:
- **Ingenuity / non-obviousness of the core idea** (key trick, surprising invariant, clever construction, unexpected transformation, "
aha" step).
- **Niche-ness / sophistication of mathematical tools** (specialized lemmas, advanced theorems, subject-specific machinery; e.g.,
olympiad-specific techniques vs routine algebra; whether it relies on tools unlikely to be known outside contest/upper-undergrad/
graduate contexts).
- **Difficulty of integrating multiple ideas** (linking several distinct concepts/lemmas; multi-layer strategy; reduction chains;
combining geometry + algebra + number theory; nontrivial case architecture as a conceptual device).

Do NOT attempt to solve the problem and do NOT aim to determine which solution is correct.

Key definition (use consistently):
"Ideological complexity" means the degree of **conceptual ingenuity, specialized tool use, and difficulty of linking multiple ideas**
in the presented solution. It is not about notation density, step count, or computation/time-to-follow.
Judge from: "How conceptually demanding is the *method/idea stack* here?" NOT: "How long does it take to parse the notation or execute
 computations?"

Complexity interpretation:
- Treat "overall complexity" as an estimate of **conceptual sophistication and methodological depth**, not time-to-follow.
- A short proof can be highly complex if it uses a deep theorem or a very non-obvious idea.
- A long proof can be low complexity if it is mostly routine expansions, bookkeeping, or standard steps.

Do NOT consider:
- **Computational/cognitive load as written**: long algebra, tedious arithmetic, dense notation, symbol tracking, or bookkeeping time.
- **Verbosity or step count** unless it reflects *genuinely* multiple conceptual ingredients (not just expanded algebra).
- Missing derivations as "hard work" unless the omission clearly hides a deep theorem/idea (again discounting routine algebraic
manipulations).
- Correctness, rigor, or whether gaps can be filled. Do not fact-check theorem applicability.

What DOES count toward ideological complexity:
1) **Ingenuity of the key move**
- Non-standard substitution or viewpoint shift (e.g., turning a Diophantine problem into a geometric/graph/invariant argument).
- Introduction of an invariant/monovariant, extremal principle, or clever construction that is not routine.
- A reduction that is conceptually subtle (e.g., "encode as generating function," "apply probabilistic method," "use compactness," etc
.).

2) **Tool sophistication / niche-ness**
- Use of advanced or specialized results (e.g., Jensen/Karamata/Muirhead in a nontrivial way; lifting exponent, LTE; Zsigmondy;
projective geometry lemmas; complex numbers/barycentric coordinates; group actions; p-adics; generating functions; spectral methods;
etc.).
- Reliance on domain-specific frameworks (e.g., functional equations classification tricks, invariant theory, combinatorial
nullstellensatz, etc.).
- The extent to which the proof requires familiarity with "IMO-level" technique stacks versus broadly taught basics.

3) **Integration burden (conceptual linking)**
- Number of distinct ideas that must be coordinated (e.g., inequality + symmetry + convexity + tangent line method).
- Multi-stage architecture (reduction → lemma → transformation → final synthesis).
- Nontrivial case splits that represent fundamentally different conceptual regimes (not mere arithmetic branching).

Edge-case handling rules (apply as needed):
1) Extremely short vs very detailed:
- Do not treat brevity as simplicity; judge the *depth of ideas/tools* used.
2) Do not judge correctness or fill gaps; judge what the solution *claims to use*.
3) If both solutions are too high-level/vague to identify tools/ideas (e.g., "clearly follows" with no method indicated), output "0" (
Tie/Indeterminate) due to insufficient evidence.
4) If one solution is computation-heavy but conceptually routine, it should rate LOW on ideological complexity even if it is hard to
follow.
5) If a solution invokes a deep theorem without explanation, you MAY count that as high ideological complexity (tool sophistication),
but do not penalize the other for not expanding computations.

You must perform a PAIRWISE COMPARISON ONLY:
- Output a single verdict: "1", "2", or "0" (Tie).
- Provide brief reasoning citing concrete features from the solutions (quote short snippets or refer to distinctive phrases like "
apply ... lemma/theorem," "consider invariant," "use generating function," etc.), but do not expand into solving steps.


Co not attempt to complete/solve the problem:
- Do not compute final answers.
- Do not re-derive results to check correctness.
- Do not introduce new math/logic beyond describing the conceptual/tooling complexity characteristics of what is already written.
```

```
- Do not "repair" a solution, propose alternatives, or add missing steps.

Tie rules (must follow; explicit):
Return "Tie/Indeterminate" if and only if at least one of the following holds:
1) **Near-equal complexity:** Neither solution is clearly more conceptually sophisticated after comparing ingenuity, tool niche-ness,
and integration.
2) **Near-identical methodology** Both solutions rely on the same core concepts, tools and strategy, with a similar method of
execution.
3) **Orthogonal tradeoffs:** One uses a deep theorem but in a single-step way, while the other uses several moderately advanced ideas
in an integrated way, and you cannot confidently rank overall ideological complexity without guessing hidden details.
4) **Insufficient evidence:** One or both solutions are too vague/underspecified to identify the conceptual toolkit or strategy.
5) **Both equally sophisticated:** Both rely on similarly niche tools and similarly non-obvious strategy.

Non-tie constraint:
Do not output Tie merely because both might be correct/incorrect/uncertain; correctness is irrelevant. Tie is only about inability to
confidently rank ideological complexity from the written text alone.

Decision procedure (follow silently; do not output these steps):
- Identify the main conceptual moves and any named tools/results in Solution 1 and Solution 2.
- Compare (a) ingenuity, (b) niche-ness/sophistication of tools, (c) integration of multiple ideas.
- Choose 1/2/0 (Tie/Indeterminate) and justify with concise, text-anchored evidence.

Output format (must follow exactly; no extra sections, no bullets, no numbering):
Solution 1 Complexity: <one concise paragraph describing the ingenuity/tools/integration complexity of Solution 1, citing 1-3 concrete
 text anchors (short quotes or distinctive phrases).>
Solution 2 Complexity: <one concise paragraph describing the ingenuity/tools/integration complexity of Solution 2, citing 1-3 concrete
 text anchors (short quotes or distinctive phrases).>
Decision Reasoning: <one concise paragraph stating the verdict (1/2/0) and explaining why, directly contrasting the biggest
ideological-complexity drivers; must reference at least one concrete feature from each solution.>
Confidence: High/Medium/Low.
Verdict: \boxed{{0  1  2}}

Now evaluate the following.

Problem:
{problem}

Solution 1:
{solution_1}

Solution 2:
{solution_2}
```

Pairwise Computational Ease Comparison Prompt We use the following prompt to compare the brute-force computational ease of different solutions based on their solving techniques.

Pairwise Computational Ease Comparison Prompt

```
You are an impartial LLM-as-a-judge. Your task is to compare Solution 1 vs Solution 2 and decide which one imposes more **notation/
computation TIME load** on a careful reader, specifically due to:
- **Notation density / symbol-tracking burden** (dense symbolic expressions; many variables/indices; summations/products; matrices/
tensors; deeply nested parentheses; frequent switching conventions; overloaded symbols).
- **Definition hygiene / referential clarity** (variables or functions used before definition; unclear domains/quantifiers; ambiguous
constraints; inconsistent naming; redefining the same symbol for different objects; unclear dependence on parameters).
- **Computation burden as written (time-to-execute/track)** (many explicit steps even if each is "simple"; long simplifications;
repeated expansions; sign/exponent/index bookkeeping; numeric work that is easy to slip on).
- **Mechanical manipulation burden** (multiple substitutions, changes of variables, rearrangements, coordinate transforms) **only
insofar as** they increase bookkeeping, symbol tracking, or explicit computation/time load.

Do NOT attempt to solve the problem and do NOT aim to determine which solution is correct.

Key definition (use consistently):
"Cognitive load" here means the mental effort and **time** required for a technically competent reader to **parse and track the
notation, definitions, and explicit computations** in the presented solution as written. It is not about correctness and not about the
 conceptual difficulty of the underlying idea.
Judge from "If I want to parse and follow what is written here line-by-line, how much time and mental effort would it take?" not "If I
 want to verify/certify every missing step, how hard would it be?"

Time Interpretation:
- Treat "overall load" as an estimate of **time-to-follow as written** for a careful, technically competent reader.
- **Many long but routine steps can be more time-consuming** than a small number of dense/clever steps.
- Count time cost coming from **step count + bookkeeping + symbol tracking**, not from conceptual depth.

Do NOT consider:
- Difficulty due to **conceptual/argument complexity** (e.g., using advanced theorems, clever insights, high-level strategy, non-
obvious ideas), **unless** it directly increases notation/definition/computation burden on the page.
```

```
- "Leaps in logic" as a correctness/rigor issue. Treat them as relevant **only** when they make the text hard to *parse* (e.g., new
symbols appear without explanation, variable roles change silently, indices/bounds are unclear).
- Whether the solution is "deep," "elegant," or "insightful."

Missing steps rule:
Do not guess the amount of work required to fill in omitted derivations unless the text itself explicitly introduces **notation/
definition/computation opacity** (e.g., new unintroduced symbols appear, variable meanings shift, indices/bounds/domains are unstated).


You must perform a pairwise comparison:
- Output a single verdict: "1", "2", or "0" (Tie).
- Provide brief reasoning that cites concrete features from the solutions (quote short snippets or refer to distinctive phrases), but
do not expand into solving steps.

STRICT NON-SOLVING RULES (must follow):
- Do not compute final answers.
- Do not re-derive results to check correctness.
- Do not introduce new math/logic beyond describing notation/definition/computation complexity characteristics of what is already
written.
- Do not "repair" a solution, propose alternatives, or add missing steps.
- Do not reward/penalize based on verbosity alone **when it is purely redundant prose**; however, **do** count verbosity that
materially increases **time-to-follow** because it adds many explicit computational/notation-tracking steps.

What to compare (qualitative rubric; no scores):
1) **Notation & symbol-tracking load (time to parse)**
- Many symbols/indices; nested expressions; heavy ∑ / ∏ notation; matrices; function composition chains.
- Overloaded symbols; notation churn; switching conventions mid-stream.
- Symbols used before being defined; unclear bounds/indices/domains.

2) **Definition hygiene / clarity of references (time lost to ambiguity)** (weigh this factor less heavily than notational and
algebraic load)
- Whether each variable/function/set is clearly introduced when first used.
- Whether constraints and quantifiers (for all/exists, domain restrictions) are explicit.
- Whether it's clear what is fixed vs varying, and what depends on what.

3) **Explicit computation & bookkeeping load (time to carry out as written)**
- Long algebraic manipulations, simplifications, expansions, casewise numeric work.
- Error-prone sign/exponent/index tracking; multiple intermediate quantities to remember.
- Multiple substitutions/rewrites that create bookkeeping overhead (regardless of conceptual motivation).
- **High step-count matters:** a long chain of straightforward arithmetic/algebra can dominate time even if each step is easy.

Edge-case handling rules (apply as needed):
1) Extremely short vs very detailed:
- A one-line claim can be low time/load to read even if conceptually deep; do not penalize conceptual depth.
- If both are too thin to compare on notation/definition/computation burden, output "0" (Tie/Indeterminate).
2) Do not judge correctness or theorem applicability; judge **notation/definition/computation time-to-follow**.
3) Do not fix apparent errors/contradictions.
4) Penalize frequent redefinition of variables, switching conventions, or overloading symbols because it increases tracking burden and
 time.
5) Many tiny trivial steps can be **more time-consuming overall** than a few dense steps; judge **overall time-to-follow**,
considering both density and total step/bookkeeping count.

Tie rules

Do not be too strict in judging one way or another if there is not a clearly dominating solution in terms of computational load. You
can call a Tie if the solutions are close enough that a reasonable reader might disagree on which is heavier, or if they have
different types of load that are hard to compare. Use the following criteria to decide whether to return **"Tie/Indeterminate"** if
you are uncertain which solution has more notation/computation load:
1) **Near-equal time/load:** After comparing notation/symbol-tracking, definition hygiene, and explicit computation/bookkeeping,
neither solution will clearly dominate the other from a human perspective. For example, if the notation introduced is not heavy to
follow, that aspect would not matter for your decision-making.
2) **Similar Solutions:** If the solution structures are for the most part similar, or the same, and the differences are mostly in
surface-level details that do not materially affect the time-to-follow (e.g., one is slightly more verbose, while the other is more
concise but uses slightly denser notation), you may judge them as a Tie.
3) **Orthogonal tradeoffs:** One solution is heavier mainly in **notation density**, while the other is heavier mainly in **long step-
by-step computation**, and you cannot confidently rank overall time-to-follow without guessing reader preferences.
4) **Insufficient evidence:** One or both solutions are too short/underspecified to assess notation/definition/computation burden **as
 written** (e.g., both are mostly high-level claims with minimal notation and no explicit computation), so a comparison would require
guessing omitted details.
5) **Both equally problematic:** Both exhibit similar levels of symbol overload, undefined variables, or dense/long manipulations such
 that a clear "more time/load" choice cannot be justified from the text alone.

Decision procedure (follow silently; do not output these steps):
- Identify the main notation/definition/computation time-load drivers in Solution 1 and in Solution 2.
- Compare which would demand more careful symbol tracking and computation/bookkeeping time to parse.
- Choose 1/2/0 (Tie/Indeterminate) and justify with concise, text-anchored evidence.

Output format (must follow exactly; no extra sections, no bullets, no numbering):
Solution 1 Load: <one concise paragraph describing the notation/definition/computation burden in Solution 1, citing 1-3 concrete text
anchors (short quotes or distinctive phrases).>
```

```
Solution 2 Load: <one concise paragraph describing the notation/definition/computation burden in Solution 2, citing 1-3 concrete text
anchors (short quotes or distinctive phrases).>
Decision Reasoning: <one concise paragraph stating the verdict (1/2/0) and explaining *why* it was chosen by directly contrasting the
biggest time/load drivers; must reference at least one concrete feature from each solution.>
Confidence: High/Medium/Low.
Verdict: \boxed{{0  1  2}}

Now evaluate the following.

Problem:
{problem}

Solution 1:
{solution_1}

Solution 2:
{solution_2}
```