LEGO-MATH: STACKING REASONING BLOCKS FOR RELIABLE AND HARDER MATH PROBLEM SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

High-quality mathematical reasoning data is essential for improving the reasoning capabilities of large language models, but obtaining such data at scale remains challenging. Existing LLM-based synthesis methods either rely on free-form prompting, which often produces invalid problems or incorrect solutions, or use rigid templates, which improve controllability but limit the diversity and difficulty of generated problems. In this work, we propose **LEGO-MATH**, a compositional framework for synthesizing mathematical reasoning problems by representing solutions as directed acyclic graphs (DAGs) and recombining them through compatible reasoning interfaces. LEGO-MATH first converts existing problem-solution pairs into reasoning DAGs, where nodes correspond to intermediate reasoning states and edges encode inferential dependencies. It then identifies compatible interfaces via edge-aware node clustering and iteratively stacks compatible DAG fragments, starting from an initial pair and extending the current composite until an adaptive stopping condition is met. Since the logical structure of the synthesized problem is inherited from verified source DAGs, correctness can be preserved by propagating only localized numerical updates along the affected downstream subgraph after each composition, reducing the role of the LLM to simple recomputation rather than open-ended generation. This structure-preserving composition enables LEGO-MATH to jointly improve *validity*, covering both problem and solution correctness, and *effectiveness*, reflected by increased problem difficulty. Extensive experiments show that LEGO-MATH produces more valid and more challenging mathematical reasoning data, and that the resulting data provides useful training signal for post-training large language models. Our code and examples of synthesized data are available at `https://anonymous.4open.science/r/LEGO-MATH-PUBLIC-5D0E`.

1 INTRODUCTION

Mathematical reasoning is a fundamental capability for large language models, requiring precise problem understanding, symbolic manipulation, and multi-step logical inference (Wang et al., 2026). Recent training paradigms have increased the demand for high-quality mathematical reasoning data (Yu et al., 2025b; An et al., 2025), yet such data is difficult to obtain at scale. Manually curated problem-solution pairs are costly to produce, as they require mathematical expertise and rigorous answer verification. As a result, LLM-based data synthesis has emerged as a primary approach for scaling math reasoning data (LI et al., 2024; Moshkov et al., 2025; Yu et al., 2025a; Manem et al., 2025; Liu et al., 2025).

Existing LLM-based synthesis methods can be broadly grouped into two lines of work, each facing distinct limitations. The first line, prompt-based synthesis (Yu et al., 2025a; 2023; Manem et al., 2025; Shah et al., 2024; Pei et al., 2025b; Li et al., 2026; Pei et al., 2025a; Zhou et al., 2024), leverages carefully designed instructions to guide LLMs in rewriting existing problems or generating new ones from scratch. This paradigm treats synthesis as a free generation process, offering no guarantee on the mathematical validity of the produced problems and the correctness of their solutions. As a result, hallucinated problems and erroneous answers are pervasive. The second line, template-based synthesis (Zhao et al., 2025b;a; Liu et al., 2025; Jung et al., 2025), constructs problems from predefined linear logic templates. Although such approaches improve controllability, their reliance on rigid templates fundamentally limits the diversity and difficulty of generated problems, making it difficult to capture the rich structural complexity exhibited by real-world mathematics. Consequently,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

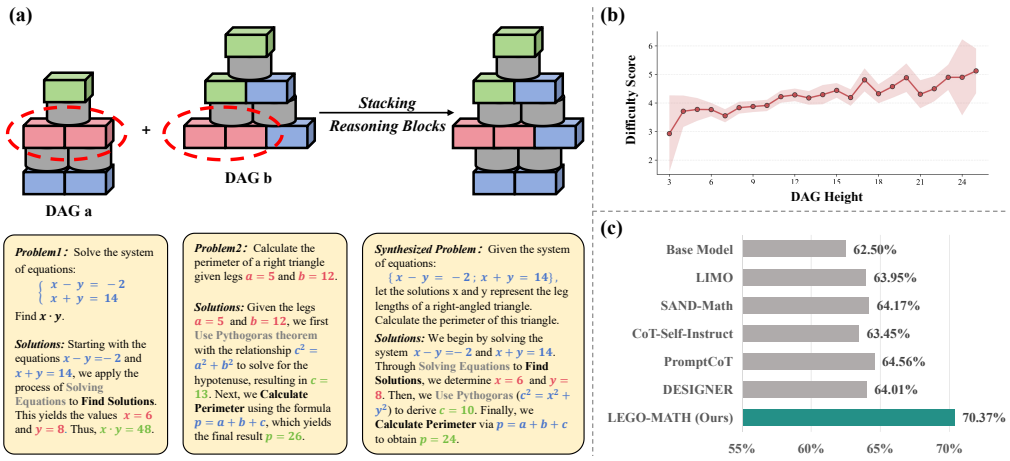


Figure 1: (a) **LEGO-based Composition.** LEGO-MATH iteratively stacks compatible DAG fragments and propagates numerical updates along affected downstream subgraphs. (b) **Problem Difficulty and DAG Height.** DAG height positively correlates with difficulty scores, suggesting that stacking DAGs taller can increase problem difficulty. (c) **Training Effectiveness.** With the same seed pool, LEGO-MATH data improve average benchmark accuracy over text-level baselines.

no existing approach is able to simultaneously ensure (i) *Validity*, covering both the correctness of synthesized problems and their corresponding solutions, and (ii) *Effectiveness*, referring to the promotion of problem difficulty in the synthesized problems.

These limitations motivate us to revisit a fundamental question: What is the appropriate unit for synthesizing math problems? We argue that a problem should not be treated as a monolithic artifact to be generated end-to-end, nor as a linear logic templates. Instead, a problem should be viewed as an assembly of reusable reasoning blocks, much like a LEGO construction, where complex structures emerge from simple, well-defined pieces connected along compatible interfaces. As shown in Figure 1 (a), this perspective reframes problem synthesis from a generative act into a compositional one, and naturally leads to two key insights that underpin our framework.

Insight 1: *The native structure of a mathematical solution is a Directed Acyclic Graph (DAG), not a chain.* Nodes denote intermediate state, and edges encode inferential dependencies. This formulation faithfully captures the multi-step dependencies and reuse patterns inherent to mathematical reasoning, where a single intermediate conclusion may rest on multiple premises, and a single intermediate quantity may feed into several downstream steps.

Insight 2: *Correctness of synthesized problems can be ensured by simple numerical recomputation.* Once two reasoning DAGs are joined at a compatible interface, the overall inferential structure of the composed problem is already well-defined by the underlying graphs themselves. What remains is merely to propagate updated values along the downstream path of the joint point. This task can reduces to simple arithmetic rather than open-ended reasoning, which confines the LLM's role to simple numerical recomputation, thereby suppressing hallucination at its source and providing a structural guarantee on the validity of the synthesized problem.

Building upon these insights, we propose **LEGO-MATH**, a compositional framework for synthesizing mathematical reasoning problems by extraction, matching, and recombining solution DAGs. At a high level, LEGO-MATH first converts existing problem-solution pairs into DAGs, where each node is reasoning step and each edge represents a reasoning action. This representation turns an unstructured chain-of-thought solution into a reusable reasoning object whose internal validity can be explicitly checked. Given a collection of such DAGs, LEGO-MATH then identifies compatible composition interfaces through edge-aware node clustering. Rather than matching nodes only by surface-level attributes such as value or type, our clustering incorporates both incoming and outgoing edges, thereby capturing the functional role of a node within its local reasoning context. Finally, LEGO-MATH synthesizes a new problem by iteratively stacking compatible DAG fragments, beginning with two DAGs, propagating numerical updates along the affected downstream subgraph after each composition, and rendering the final corrected graph after adaptive stopping. Importantly, the logical

structure of the composed solution is inherited from the verified source DAGs, while the LLM is only asked to perform localized numerical recomputation under fixed dependencies. In this way, LEGO-MATH transforms data synthesis from unconstrained generation into structure-preserving composition, simultaneously improving controllability, correctness, and problem complexity. We summarize the contributions as following:

- We introduce a DAG-based formulation of mathematical problem synthesis, which exposes the latent dependency structure of solutions and enables reasoning blocks to be reused compositionally.
- We develop **LEGO-MATH**, a structure-preserving synthesis framework that verifies reasoning DAGs, composes them via edge-aware interfaces, and updates only downstream numerical values, enabling harder problem synthesis while preserving correctness by construction.
- Extensive experiments show that **LEGO-MATH** produces valid, correct, and more challenging mathematical reasoning data compared with prompt-based and template-based synthesis baselines.

2 Related Work

Math datasets. Math training corpora emphasize either scale or quality. Large-scale corpora such as NuminaMath (LI et al., 2024), MathInstruct (Yue et al., 2023), DeepMath (He et al., 2025), OpenMathReasoning (Moshkov et al., 2025), OpenMathInstruct-2 (Toshniwal et al., 2025), and ScaleQuest (Ding et al., 2025) collect or synthesize diverse math problems with CoT (Wei et al., 2022) or PoT (Chen et al., 2022) rationales for math SFT. LIMO (Ye et al., 2025), s1 (Muennighoff et al., 2025), and DART-Math (Tong et al., 2024) show that smaller or selectively constructed datasets can also induce strong reasoning, highlighting the importance of data quality (Zhao et al., 2025c). LEGO-MATH targets this quality dimension by composing verified reasoning structures into correct, more challenging training problems.

Mathematical reasoning benchmarks have progressed from grade-school problems to competition-level and expert-level challenges. GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) remain foundational benchmarks for grade-school and high-school competition mathematics. As frontier models increasingly saturate these benchmarks, evaluations have shifted toward harder settings: real competitions such as AIME 2025 (Zhang & Math-AI, 2025) and IMO 2025 (LMMS Lab, 2025), olympiad-style suites such as OlympiadBench (He et al., 2024) and Omni-MATH (Gao et al., 2024), and expert-authored or university-level benchmarks such as FrontierMath (Glazer et al., 2024), MathBench (Liu et al., 2024), MathOdyssey (Fang et al., 2025), U-MATH (Chernyshev et al., 2024), and CompMath-MCQ (Raimondi et al., 2026). These benchmarks emphasize difficulty and reasoning reliability. We primarily evaluate on AIME and AMC competition benchmarks.

Math data synthesis. To reduce the cost of human annotation, recent work has explored synthesizing mathematical reasoning data with LLMs (Seegmiller et al., 2025). Text-oriented methods use self-instruction, problem evolution, augmentation, instruction fusion, or specialized generators to create training examples (Yu et al., 2025a; 2023; Manem et al., 2025; Shah et al., 2024; Pei et al., 2025b; Li et al., 2026; Pei et al., 2025a; Zhou et al., 2024; Li et al., 2024; You et al., 2024). MathFusion specifically combines related problems through sequential, parallel, and conditional fusion (Pei et al., 2025b). Scheherazade (Miner et al., 2024) and the compositional-math study of Xiao & Zhao (2025) construct multi-problem instances primarily to evaluate chained or compositional reasoning. Other methods guide synthesis with construction rationales, design logics, or gradient-space diversity (Zhao et al., 2025b;a; Liu et al., 2025; Jung et al., 2025); MathScale and CogAtom use concept graphs or recombined cognitive atoms (Tang et al., 2024; Chen et al., 2025). Neuro-symbolic data generation uses math solvers to help maintain validity while varying problems (Li et al., 2024); Lean 4 and LeanDojo provide complementary theorem-proving infrastructure (Moura & Ullrich, 2021; Yang et al., 2023). LEGO-MATH composes verified solution DAGs at compatible interfaces and propagates numerical changes downstream, aiming to improve difficulty and correctness by construction beyond post-hoc filtering or unconstrained generation.

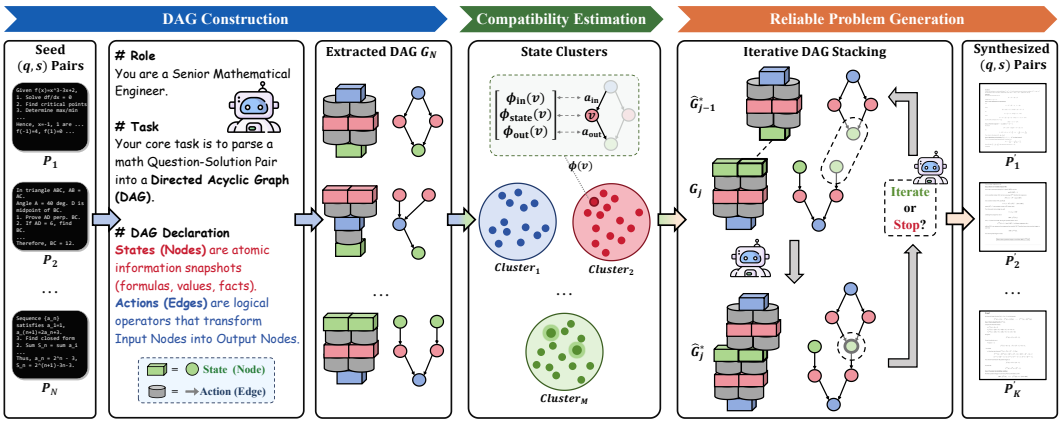


Figure 2: **LEGO-MATH overview.** Public problem–solution pairs are converted into verified solution DAGs, matched through compatible bridge states, and iteratively composed. Starting from a compatible pair, LEGO-MATH selects further extensions by marginal synthesis gain and future extensibility, applying constrained value propagation after each extension. The final corrected composite is rendered after adaptive stopping.

3 METHOD

3.1 PRELIMINARIES

Problem formulation. We study the problem of synthesizing new mathematical question–solution pairs from existing public data. Let the seed dataset be $\mathcal{D} = \{(q_i, s_i)\}_{i=1}^{N}$, where q_i denotes a mathematical question, s_i denotes its corresponding solution, and N is the number of seed pairs. The goal of LEGO-MATH is to construct a synthesized dataset $\mathcal{D}' = \{(q'_j, s'_j)\}_{j=1}^{M}$, where each pair (q'_j, s'_j) is derived from existing solution structures rather than unrestricted prompt-based rewriting.

Synthesis objectives. Following the motivation in the introduction, we consider two desiderata: *validity* and *effectiveness*. *Validity* requires that the synthesized question q'_j is mathematically well-defined and its solution s'_j is correct and internally consistent. *Effectiveness* refers to promoting problem difficulty, rather than producing trivial paraphrases or surface-level variants of seed questions. LEGO-MATH addresses these objectives by composing verified solution dependencies: verified fragments support validity, while cross-DAG recombination creates richer reasoning paths.

3.2 METHOD OVERVIEW

Figure 2 gives an overview of LEGO-MATH. We formulate synthesis as a structured composition problem over solution DAGs, with three technical stages: DAG construction, compatibility estimation, and reliable problem generation. At a high level, the pipeline is

$$(q, s) \xrightarrow{\mathcal{E}} G \xrightarrow{\mathcal{C}} (G_a, G_b, v_a, v_b) \xrightarrow{\text{iterate } \mathcal{S},\text{Prop}} \widehat{G}^{\star}_{T} \xrightarrow{\mathcal{R}} (q', s'). \tag{1}$$

Here, $\mathcal{E}$ denotes DAG extraction, $\mathcal{C}$ denotes compatibility estimation, $\mathcal{S}$ denotes stacking, and Prop denotes constrained value propagation. Starting from an initial bridge-state tuple, synthesis alternates stacking and value propagation, reusing the compatibility mechanism to retrieve further extensions until an adaptive stopping condition is met. The final operator $\mathcal{R}$ only verbalizes the final corrected composite $\widehat{G}^{\star}_{T}$ into a natural-language question–solution pair, where T denotes the adaptively determined number of composed source DAGs. The key insight is to synthesize over DAG-structured solution dependencies, where verified fragments can be recomposed through compatible bridge states while preserving topological validity. Thus, LEGO-MATH reduces the LLM's role from unrestricted problem generation to bounded parsing, verification, and value propagation.

3.3 DAG CONSTRUCTION

LEGO-MATH converts each seed question–solution pair into a validated solution DAG. Extraction uses ontology-guided semantic tags, followed by graph verification to ensure reliable composition.

Given a seed pair $(q_i, s_i) \in \mathcal{D}$ from domain d, we instantiate the extraction operator in Eq. (1) as

$$G_i = \mathcal{E}_{\Omega_d}(q_i, s_i), \qquad G_i = (V_i, A_i), \tag{2}$$

where Ω_d denotes the semantic tag space for domain d, V_i is the set of intermediate mathematical states, and A_i is the set of directed dependency actions. Each node $v \in V_i$ represents a state appearing in or implied by the solution, such as an intermediate equation, variable assignment, derived value, constraint, or condition. Each action $(u, v) \in A_i$ indicates that state u is used to derive state v.

To make the extracted graphs semantically stable across questions, we assign each state node a set of ontology-guided semantic tags. The tags describe three aspects of a mathematical state: *subject*, *attribute*, and *representation*. For example, a node may be tagged by what mathematical object it refers to, what property is being specified, and how the state is expressed in symbolic form. During extraction, the LLM is required to annotate each node with these semantic tags, following the domain-specific tag space defined for `Algebra`, `Number Theory`, and `Combinatorics`. This prevents the DAG from becoming a loose collection of natural-language fragments and provides a shared schema for cross-problem matching in Section 3.4. Full tag definitions appear in Appendix C.

LEGO-MATH validates each extracted graph for acyclicity, tag consistency, and step-level mathematical correctness. Verification follows topological order, checking downstream states only after their required upstream states are established. This yields a verified DAG pool $\mathcal{G}$.

3.4 COMPATIBILITY ESTIMATION

Given the verified DAG pool $\mathcal{G}$ from Section 3.3, LEGO-MATH identifies compatible bridge states for reliable cross-problem composition. A key challenge is that compatibility cannot be determined by the isolated node content alone. Two states may have the same value, type, or symbolic form, but play different roles in their respective solutions. For example, two nodes may both state "$x = 5$", yet one is an assumption while the other is derived as the positive root of $x^2 - 25 = 0$. LEGO-MATH therefore matches nodes by *functional substitutability*, whether they are produced and used in compatible ways rather than by isolated surface similarity.

For each node v in a verified DAG, we construct an edge-aware state representation:

$$\phi(v) = \big[\phi_{\text{state}}(v); \phi_{\text{in}}(v); \phi_{\text{out}}(v)\big], \tag{3}$$

where $\phi_{\text{state}}(v)$ encodes the node content and its semantic tags, $\phi_{\text{in}}(v)$ summarizes the incoming dependency context that derives v, and $\phi_{\text{out}}(v)$ summarizes the outgoing dependency context in which v is used. The incoming context helps distinguish whether a state is assumed, computed, transformed, or constrained, while the outgoing context captures what later reasoning depends on that state. Using $\phi(v)$, LEGO-MATH clusters nodes across the verified DAG pool. Nodes in the same cluster are treated as candidate bridge states because they share similar mathematical roles under both node-level and edge-level contexts. The resulting bridge-state pool is written as

$$\mathcal{B} = \mathcal{C}(\mathcal{G}) = \{(G_a, G_b, v_a, v_b) \mid v_a \sim_\phi v_b\}, \tag{4}$$

where $G_a, G_b \in \mathcal{G}$ are two distinct verified DAGs, $v_a \in G_a$ and $v_b \in G_b$ are candidate bridge states, $v_a \sim_\phi v_b$ indicates that the two nodes are assigned to a compatible edge-aware cluster. The output of this stage is the bridge-state pool $\mathcal{B}$, which provides the initial stacking points for the next stage. Subsequent extensions reuse the same edge-aware compatibility mechanism on the current corrected composite and candidate source DAGs, with state representations evaluated using their current node contents and dependency contexts.

3.5 RELIABLE PROBLEM GENERATION

The final technical stage of LEGO-MATH synthesizes new question–solution pairs by composing verified solution fragments. Given a bridge-state tuple $(G_a, G_b, v_a, v_b) \in \mathcal{B}$ from Section 3.4, the goal is to use the upstream derivation of v_a in G_a as the input condition for the downstream continuation

of v_b in G_b. In this A-to-B stacking pattern, the first DAG derives the bridge state, and the second provides a verified continuation conditioned on a compatible state.

Let $G_a^{\preceq v_a}$ denote the upstream subgraph of G_a that contains v_a and all states required to derive it. Similarly, let $G_b^{\succeq v_b}$ denote the downstream subgraph of G_b that contains v_b and all states depending on it. The stacking operation is defined as

$$\mathcal{S}(G_a, G_b, v_a, v_b) = G_a^{\preceq v_a} \oplus_{v_a \to v_b} G_b^{\succeq v_b}, \tag{5}$$

where $\oplus_{v_a \to v_b}$ replaces the original input state v_b with the output state v_a and redirects the downstream dependencies of v_b to the value produced by v_a. Since the new dependency only flows from the upstream fragment to the downstream fragment, the stacked graph preserves a valid topological order and avoids circular reasoning. Therefore, the main remaining risk is not the logical structure of the composed solution, but the numerical consistency of downstream states after replacing v_b with v_a.

Denote the stacked graph by $G^\star = \mathcal{S}(G_a, G_b, v_a, v_b)$. LEGO-MATH then performs constrained value propagation. Starting from the bridge state v_a, the system traverses $G^\star$ in topological order and recomputes only the affected downstream numerical fields:

$$\widehat{G}^\star = \mathrm{Prop}(G^\star; v_a). \tag{6}$$

The semantic tags, dependency actions, and inference operations are kept fixed, so $\widehat{G}^\star$ denotes the same reasoning structure with corrected numerical values. Thus, the LLM is restricted to local arithmetic or symbolic recalculation under a fixed reasoning structure.

This stacking and correction process is applied iteratively. Let $G_j^\star$ denote a stacked graph comprising fragments from j source DAGs, and let $\widehat{G}_j^\star$ denote its value-corrected version. Starting from the initial pair, composition and correction proceed as

$$G_j^\star = \begin{cases} \mathcal{S}(G_a, G_b, v_a, v_b), & j = 2, \\ \mathcal{S}(\widehat{G}_{j-1}^\star, G_j, u_{j-1}, v_j), & 2 < j \le T, \end{cases} \qquad \widehat{G}_j^\star = \begin{cases} \mathrm{Prop}(G_j^\star; v_a), & j = 2, \\ \mathrm{Prop}(G_j^\star; u_{j-1}), & 2 < j \le T. \end{cases} \tag{7}$$

Here, G_j is the j-th selected source DAG, and u_{j-1} and v_j are compatible bridge states in the current composite and the new DAG, retrieved using the same edge-aware compatibility mechanism from Section 3.4. Each extension is followed by constrained value propagation from the new bridge state before the result is used in the next composition.

Candidate extensions must satisfy bridge compatibility and source-DAG quality constraints. Among feasible candidates, LEGO-MATH favors greater marginal synthesis gain, reflecting novelty, topology, source diversity, difficulty, and length compatibility, while considering future extensibility based on available subsequent compatible extensions. Iteration stops when no feasible extension remains, the best candidate offers insufficient marginal gain, or a prescribed composition-size or length budget is reached, determining T adaptively. Only then is the final pair rendered:

$$(q', s') = \mathcal{R}(\widehat{G}_T^\star), \tag{8}$$

where $\mathcal{R}$ denotes the renderer that verbalizes the final corrected graph into a natural-language question–solution pair. Pairwise stacking is the initialization and the $T = 2$ special case of this process; iterative extension progressively grows the reasoning structure while preserving local validity.

4 Experiments

4.1 Setup

Base models and training. We apply GRPO (Shao et al., 2024) to released `Qwen3-4B-Instruct-2507` without prior task-specific SFT or RL: accuracy reward with `math_verify` for numeric answers, temperature 0.6, top-p 0.95, 8 rollouts per prompt, and 1 epoch on 8× H20 GPUs. `Phi-4-mini-instruct` and `Llama-3.2-3B-Instruct` use the same data budget and protocol (full hyperparameters: Appendix A).

Evaluation. We average final-answer accuracy over 8 independent runs on 224 AIME24 (Zhang & Math-AI, 2024), AIME25 (Zhang & Math-AI, 2025), AIME26 (Zhang & Math-AI, 2026), and AMC22/23/24 problems, retaining training temperature/top-p and weighting by benchmark size.

Table 1: **GRPO training results on competition mathematics benchmarks.** Accuracy (%): mean of 8 runs; *Avg.*: weighted over 224 problems in six benchmarks (sample SDs: Appendix Table 7). *# Train*: corpus size, with 800 LIMO examples plus 800 per synthesis method. *w/o DAG*: direct pair composition and rejection sampling without DAG construction. Best in **bold**; next lower result underlined.

Setting	# Train	AIME24	AIME25	AIME26	AMC22	AMC23	AMC24	Avg.
Baselines								
Qwen3-4B-Instruct-2507	0	51.67	35.83	37.08	78.49	81.52	69.72	62.50
LIMO	800	52.50	36.67	44.17	78.49	80.16	72.50	63.95
Text-level synthesis								
SAND-Math	800+800	53.33 (+0.83)	37.50 (+0.83)	44.58 (+0.41)	77.03 (-1.46)	84.24 (+4.08)	69.44 (-3.06)	64.17 (+0.22)
CoT-Self-Instruct	800+800	52.50 (±0.00)	32.50 (-4.17)	41.67 (-2.50)	77.33 (-1.16)	83.42 (+3.26)	72.22 (-0.28)	63.45 (-0.50)
Intermediate-representation synthesis								
PromptCoT	800+800	50.00 (-2.50)	39.17 (+2.50)	43.33 (-0.84)	79.94 (+1.45)	80.98 (+0.82)	73.89 (+1.39)	64.56 (+0.61)
DESIGNER	800+800	51.25 (-1.25)	34.58 (-2.09)	41.67 (-2.50)	78.49 (±0.00)	83.97 (+3.81)	72.78 (+0.28)	64.01 (+0.06)
Our Method								
LEGO-MATH (w/o DAG)	800+800	56.67 (+4.17)	**43.75** (+7.08)	50.83 (+6.66)	73.26 (-5.23)	85.33 (+5.17)	77.78 (+5.28)	67.47 (+3.52)
LEGO-MATH (ours)	800+800	**58.75** (+6.25)	**43.75** (+7.08)	**52.08** (+7.91)	**84.30** (+5.81)	**86.68** (+6.52)	**78.06** (+5.56)	**70.37** (+6.42)

Table 2: **Cross-family GRPO training results.** Data budgets and evaluation follow Table 1. Best within each backbone in **bold**; standard deviations appear in Appendix Table 8.

Setting	# Train	AIME24	AIME25	AIME26	AMC22	AMC23	AMC24	Avg.
Phi-4-mini-instruct								
Base Model	0	5.00	4.58	5.00	21.51	23.37	20.28	14.96
LIMO	800	6.67	**5.42**	**5.42**	23.84	25.27	21.94	16.52
LEGO-MATH	800+800	**8.33**	**5.42**	**5.42**	**24.71**	**29.08**	**25.28**	**18.36**
Llama-3.2-3B-Instruct								
Base Model	0	4.58	0.00	0.42	9.59	15.22	8.06	7.25
LIMO	800	4.58	0.83	0.83	13.66	**16.58**	10.28	8.93
LEGO-MATH	800+800	**5.00**	**1.25**	**2.92**	**15.70**	**16.58**	**10.83**	**9.82**

Baselines. Baselines are human-curated LIMO (Ye et al., 2025) (800 examples), text-level CoT-Self-Instruct (Yu et al., 2025a) and SAND-Math (Manem et al., 2025), and intermediate-representation PromptCoT (Zhao et al., 2025b) and DESIGNER (Liu et al., 2025), using original recipes (Appendices A and B).

Seed data. Each method synthesizes 800 problems from 2640 historical AMC/AIME seeds in NuminaMath-CoT, retained after exact, near-duplicate, and embedding-based filtering against all benchmark instances.

4.2 Training Value

Each method uses 800 synthetic + 800 LIMO examples (1600 total). Tables 1–2 report GRPO results for Qwen3 (including *w/o DAG*) and two other model families.

(a) LEGO-MATH outperforms full synthesis baselines on all six benchmarks. Overall accuracy is 70.37%: +6.42 points over LIMO, +7.87 over the released checkpoint, and +5.81 over the best

Table 3: **Cross-method quality comparison.** *V* denotes validity (%) and *E* denotes effectiveness (%); higher is better for both. Effectiveness is conditional on validity. Per-column best in bold.

	Algebra		Number Theory		Combinatorics		Overall	
Method	V ↑	E ↑	V ↑	E ↑	V ↑	E ↑	V ↑	E ↑
SAND-Math	22.19	3.97	17.39	4.35	16.84	4.21	18.81	4.18
PromptCoT	21.23	3.77	34.88	6.98	31.82	6.82	29.31	5.86
DESIGNER	59.38	2.46	50.91	0.00	53.33	1.90	54.54	1.45
CoT-Self-Instruct	75.15	0.31	71.15	1.92	63.04	2.17	69.78	1.47
LEGO-MATH (ours)	**86.84**	**7.89**	**77.78**	**19.44**	**81.08**	**16.22**	**81.90**	**14.52**

full baseline, PromptCoT (64.56%). Including direct composition, LEGO-MATH leads on five benchmarks and ties on AIME25.

(b) The advantage grows with problem difficulty. Margins over the best full baselines are $5.42/4.58/7.50$ points on AIME24/25/26 and $4.36/2.44/4.17$ on AMC22/23/24. The 15.00-point peak over the released checkpoint on AIME26 suggests greater benefit on harder problems.

(c) Structure-preserving synthesis improves post-training results. Direct pair composition with strong-model rejection sampling (*w/o DAG*) scores 67.47% ($+3.52$ over LIMO). LEGO-MATH gains 2.90 points with no benchmark losses, supporting DAG composition, compatible-state matching, and localized recomputation.

LEGO-MATH beats LIMO in nine of twelve cross-family comparisons, tying three (Table 2): Phi-4-mini improves from 16.52% to 18.36% and Llama-3.2-3B from 8.93% to 9.82%. This supports GRPO transfer across 3B–4B families; larger models and other post-training methods remain untested.

4.3 Quality of Synthesized Data

Seeds are solved by Qwen3-Max. *Validity* requires internally consistent problems with correct solutions; *effectiveness* measures difficulty relative to this solver: the fraction of valid problems it cannot solve. Tables 3–4 compare original and shared generation/verification models.

(a) LEGO-MATH leads on both metrics across domains. Validity is 81.90% and effectiveness 14.52%, versus baseline bests of 69.78% and 5.86% (Table 3). Effectiveness peaks in Number Theory (19.44%), then Combinatorics (16.22%).

(b) LEGO-MATH overcomes the baselines' validity–effectiveness trade-off. SAND-Math/PromptCoT yield $18.81/29.31\%$ validity and $4.18/5.86\%$ effectiveness; DESIGNER/CoT-Self-Instruct have higher validity ($54.54/69.78\%$) but lower effectiveness ($1.45/1.47\%$). LEGO-MATH leads all four on both, supporting verified DAG composition for valid, harder problems.

With shared Gemini-3-Flash synthesis and GPT-5.2 verification (Table 4), PromptCoT approaches LEGO-MATH's validity (81.72% vs. 81.90%) but trails by 10.78 effectiveness points (3.74% vs. 14.52%); DESIGNER scores 79.57% and 1.17%. Thus, the effectiveness advantage persists at controlled model strength and comparable validity.

4.4 Ablation Study

Removing the DAG. Direct composition with strong-model rejection sampling lowers both quality metrics (Figure 3) and training accuracy versus LEGO-MATH, while outperforming LIMO (Table 1), supporting the DAG pipeline's quality and training benefits.

4.5 Analysis

Human Assessment. Three experts independently rate shared problems (Figure 4) for *logical consistency* (well-formedness without contradictions), *fusion elegance* (natural reasoning integration), and *innovation* (non-trivial reasoning beyond surface rewriting). Agreement is strong (Table 5): ranking $W = 0.704$–0.772; averaged-rating ICC(3,3)$= 0.816$–0.881, allowing scale differences

Table 4: **Quality comparison under a unified configuration.** Shared synthesis: Gemini-3-Flash; verification: GPT-5.2. V: validity (%); E: effectiveness (%). Best bold; second-best underlined.

Method	V ↑	E ↑
CoT-Self-Instruct	55.22	2.15
DESIGNER	79.57	1.17
PromptCoT	<u>81.72</u>	<u>3.74</u>
SAND-Math	55.20	1.14
LEGO-MATH (ours)	**81.90**	**14.52**

Table 5: **Inter-annotator agreement for human evaluation.** Three experts: Kendall's W measures ranking consistency; ICC(3,3), reliability of averaged ratings. All W: $p < 0.001$.

Dimension	Kendall's W	ICC(3,3), 95% CI
Logical consistency	0.713	0.852 [0.707, 0.925]
Fusion elegance	0.704	0.816 [0.588, 0.906]
Innovation	0.772	0.881 [0.769, 0.933]

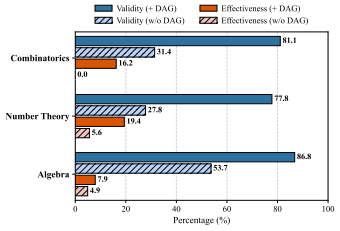


Figure 3: **Ablation on the DAG representation.** Validity and effectiveness with and without the reasoning DAG.

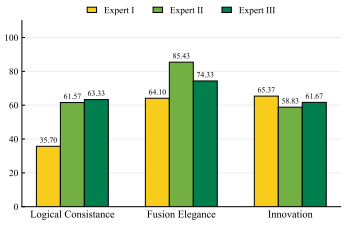


Figure 4: **Human assessment of synthesized problems.** Expert ratings on logical consistency, fusion elegance, and innovation.

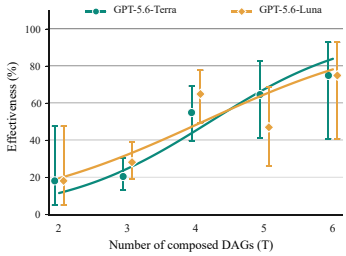


Figure 5: **Effectiveness versus DAG count.** One minus solver pass rate, with logistic fits and 95% Wilson confidence intervals.

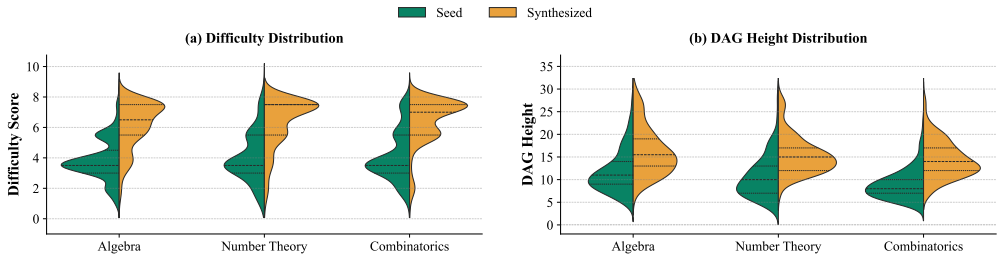


Figure 6: **(a) Difficulty distribution and (b) DAG height distribution.** Seed and synthesized problems are compared across algebra, number theory, and combinatorics.

among the three fixed experts. Lower, more variable logical-consistency scores indicate remaining subtle inconsistencies.

Difficulty and DAG height. Increased difficulty and DAG height over seeds (Figure 6) suggest longer reasoning chains; height is an empirical depth proxy, not a complete difficulty measure.

Solver resistance and DAG count. For GPT-5.6-Luna and GPT-5.6-Terra, effectiveness (one minus pass rate) rises from about 18% at two DAGs to 75% at six (Figure 5). Luna's drop from 65% at four to 47% at five shows non-monotonicity. Only eight six-DAG problems yield wide confidence intervals. This association is not causal evidence and is separate from Qwen3-Max results (Tables 3–4).

5 Conclusion

We proposed **LEGO-MATH**, which synthesizes mathematical reasoning data by composing verified solution DAGs through compatible nodes, increasing problem difficulty while preserving logical consistency. Experiments show higher validity, effectiveness, and downstream training benefits than text-level and intermediate-representation baselines. These results support solution-structure-guided composition for building high-quality mathematical reasoning data.

AI USE STATEMENT

Our research pipeline (Sections 3 and 4) uses Gemini 3 Flash and GPT-5.2 for DAG extraction, problem synthesis, and verification, and Qwen3-Max and the other reported solvers for difficulty assessment and evaluation. Appendix A details model roles and configurations; Appendix G provides the prompts. Synthesized outputs undergo structural and model-based checks, with sampled problems independently assessed by three experts (Section 4.5). These checks provide empirical validation rather than formal correctness guarantees. Section 4.1 identifies the models fine-tuned and evaluated.

We also used generative AI as an auxiliary manuscript reviewer to flag possible issues in clarity, presentation, and the interpretation of results; the authors independently evaluated this feedback and wrote all revisions. We did not use generative AI to draft sections, translate the manuscript, implement algorithms, create figures, or design experiments. The authors take responsibility for the manuscript, its claims, and accompanying research artifacts.

REPRODUCIBILITY STATEMENT

Section 3 specifies the DAG construction, compatibility estimation, and composition procedure; Appendix E gives the full pipeline, and Appendices C–D define its domain-specific annotations and actions. Section 4.1 describes the data sources, benchmark filtering, baselines, and evaluation protocol. Appendix A records the training and inference configurations and model roles, Appendix G contains the synthesis prompts, and Appendix F reports variation across runs. Dataset and model provenance are listed in Appendix B; anonymous code and example synthesized data are linked in the abstract.

REFERENCES

Chenxin An, Zhihui Xie, Xiaonan Li, Ming Zhong, Shansan Gong, Lei Li, Jun Zhang, Jingjing Xu, and Lingpeng Kong. Long chain-of-thought fine-tuning via understanding-to-reasoning transition. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 34518–34534, 2025.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Zhuofan Chen, Jiyuan He, Yichi Zhang, Xing Hu, Haoxing Wen, Jun Bai, and Wenge Rong. CogAtom: From cognitive atoms to olympiad-level mathematical reasoning in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 24108–24125. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.1309. URL https://aclanthology.org/2025.findings-emnlp.1309/.

Konstantin Chernyshev, Vitaliy Polshkov, Ekaterina Artemova, Alex Myasnikov, Vlad Stepanov, Alexei Miasnikov, and Sergei Tilga. U-MATH: A university-level benchmark for evaluating mathematical skills in LLMs. *arXiv preprint arXiv:2412.03205*, 2024.

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

Yuyang Ding, Xinyu Shi, Xiaobo Liang, Juntao Li, Zhaopeng Tu, Qiaoming Zhu, and Min Zhang. Unleashing LLM reasoning capability via scalable question synthesis from scratch. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13414–13438. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.658. URL https://aclanthology.org/2025.acl-long.658/.

Meng Fang, Xiangpeng Wan, Fei Lu, Fei Xing, and Kai Zou. MathOdyssey: Benchmarking mathematical problem-solving skills in large language models using odyssey math data. *Scientific data*, 12(1):1392, 2025.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, et al. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*, 2024.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, et al. FrontierMath: A benchmark for evaluating advanced mathematical reasoning in AI. *arXiv preprint arXiv:2411.04872*, 2024.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. DeepMath-103K: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.

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

Jaehun Jung, Seungju Han, Ximing Lu, Skyler Hallinan, David Acuna, Shrimai Prabhumoye, Mostafa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Yejin Choi. Prismatic synthesis: Gradient-based data diversification boosts generalization in LLM reasoning. *arXiv preprint arXiv:2505.20161*, 2025.

Chengpeng Li, Zheng Yuan, Hongyi Yuan, Guanting Dong, Keming Lu, Jiancan Wu, Chuanqi Tan, Xiang Wang, and Chang Zhou. MuggleMath: Assessing the impact of query and response augmentation on math reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10230–10258. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.551. URL https://aclanthology.org/2024.acl-long.551/.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath-CoT. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/AI-MO/NuminaMath-CoT.

Xuchen Li, Jing Chen, Xuzhao Li, Hao Liang, Xiaohuan Zhou, Taifeng Wang, and Wentao Zhang. MathMixup: Boosting LLM mathematical reasoning with difficulty-controllable data synthesis and curriculum learning. *arXiv preprint arXiv:2601.17006*, 2026.

Zenan Li, Zhi Zhou, Yuan Yao, Yu-Feng Li, Chun Cao, Fan Yang, Xian Zhang, and Xiaoxing Ma. Neuro-symbolic data generation for math reasoning. *Advances in Neural Information Processing Systems*, 37: 23488–23515, 2024.

Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. MathBench: Evaluating the theory and application proficiency of LLMs with a hierarchical mathematics benchmark. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6884–6915, 2024.

Weize Liu, Yongchi Zhao, Yijia Luo, Mingyu Xu, Jiaheng Liu, Yanan Li, Xiguo Hu, Zhiqi Bai, Yuchi Xu, Wenbo Su, et al. DESIGNER: Design-logic-guided multidisciplinary data synthesis for LLM reasoning. *arXiv preprint arXiv:2508.12726*, 2025.

LMMS Lab. IMO 2025 problems dataset, 2025. URL https://huggingface.co/datasets/lmms-lab/imo-2025.

Chaitanya Manem, Pratik Prabhanjan Brahma, Prakamya Mishra, Zicheng Liu, and Emad Barsoum. SAND-Math: Using LLMs to generate novel, difficult and useful mathematics questions and answers. *arXiv preprint arXiv:2507.20527*, 2025.

Stephen Miner, Yoshiki Takashima, Simeng Han, Sam Kouteili, Ferhat Erata, Ruzica Piskac, and Scott J Shapiro. Scheherazade: Evaluating chain-of-thought math reasoning in LLMs with chain-of-problems. *arXiv preprint arXiv:2410.00151*, 2024.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. AIMO-2 winning solution: Building state-of-the-art mathematical reasoning models with OpenMathReasoning dataset. *arXiv preprint arXiv:2504.16891*, 2025.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *International Conference on Automated Deduction*, pp. 625–635. Springer, 2021.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321, 2025.

Qizhi Pei, Zhuoshi Pan, Honglin Lin, Xin Gao, Yu Li, Zinan Tang, Conghui He, Rui Yan, and Lijun Wu. ScaleDiff: Scaling difficult problems for advanced mathematical reasoning. *arXiv preprint arXiv:2509.21070*, 2025a.

Qizhi Pei, Lijun Wu, Zhuoshi Pan, Yu Li, Honglin Lin, Chenlin Ming, Xin Gao, Conghui He, and Rui Yan. MathFusion: Enhancing mathematical problem-solving of LLM through instruction fusion. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7400–7420, 2025b.

Bianca Raimondi, Francesco Pivi, Davide Evangelista, and Maurizio Gabbrielli. The CompMath-MCQ dataset: Are llms ready for higher-level math? *arXiv preprint arXiv:2603.03334*, 2026.

Parker Seegmiller, Kartik Mehta, Soumya Saha, Chenyang Tao, Shereen Oraby, Arpit Gupta, Tagyoung Chung, Mohit Bansal, and Nanyun Peng. FLAMES: Improving LLM math reasoning via a fine-grained analysis of the data synthesis pipeline. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 24746–24766. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.1346. URL https://aclanthology.org/2025.findings-emnlp.1346/.

Vedant Shah, Dingli Yu, Kaifeng Lyu, Simon Park, Jiatong Yu, Yinghui He, Nan Rosemary Ke, Michael Mozer, Yoshua Bengio, Sanjeev Arora, et al. AI-assisted generation of difficult math questions. *arXiv preprint arXiv:2407.21009*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. MathScale: Scaling instruction tuning for mathematical reasoning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47885–47900. PMLR, 2024. URL https://proceedings.mlr.press/v235/tang24k.html.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. DART-Math: Difficulty-aware rejection tuning for mathematical problem-solving. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0251. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/0ef1afa0daa888d695dcd5e9513bafa3-Abstract-Conference.html.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. OpenMathInstruct-2: Accelerating ai for math with massive open-source instruction data. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/302ce0673c00aee2cf84bb43d0117553-Abstract-Conference.html.

Peng-Yuan Wang, Tian-Shuo Liu, Chenyang Wang, Ziniu Li, Yidi Wang, Shu Yan, Chengxing Jia, Xu-Hui Liu, Xinwei Chen, Jiacheng Xu, et al. A survey on large language models for mathematical reasoning. *ACM Computing Surveys*, 58(8):1–35, 2026.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Xisheng Xiao and Hanlin Zhao. From A and B to A+B: Can large language models solve compositional math problems? In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13057–13078, 2025.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J Prenger, and Animashree Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. *Advances in Neural Information Processing Systems*, 36:21573–21612, 2023.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. LIMO: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

Weihao You, Shuo Yin, Xudong Zhao, Zhilong Ji, Guoqiang Zhong, and Jinfeng Bai. MuMath: Multi-perspective data augmentation for mathematical reasoning in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2932–2958. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-naacl.185. URL https://aclanthology.org/2024.findings-naacl.185/.

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

Ping Yu, Jack Lanchantin, Tianlu Wang, Weizhe Yuan, Olga Golovneva, Ilia Kulikov, Sainbayar Sukhbaatar, Jason Weston, and Jing Xu. CoT-Self-Instruct: Building high-quality synthetic prompts for reasoning and non-reasoning tasks. *arXiv preprint arXiv:2507.23751*, 2025a.

Yiyao Yu, Yuxiang Zhang, Dongdong Zhang, Xiao Liang, Hengyuan Zhang, Xingxing Zhang, Mahmoud Khademi, Hany Hassan Awadalla, Junjie Wang, Yujiu Yang, et al. Chain-of-reasoning: Towards unified mathematical reasoning in large language models via a multi-paradigm perspective. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24914–24937, 2025b.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MAmmoTH: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (AIME) 2024, 2024. URL https://huggingface.co/datasets/math-ai/aime24.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (AIME) 2025, 2025. URL https://huggingface.co/datasets/math-ai/aime25.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (AIME) 2026, 2026. URL https://huggingface.co/datasets/math-ai/aime26.

Xueliang Zhao, Wei Wu, Jian Guan, Zhuocheng Gong, and Lingpeng Kong. PromptCoT 2.0: Scaling prompt synthesis for large language model reasoning. *arXiv preprint arXiv:2509.19894*, 2025a.

Xueliang Zhao, Wei Wu, Jian Guan, and Lingpeng Kong. PromptCoT: Synthesizing olympiad-level problems for mathematical reasoning in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18167–18188, 2025b.

Yike Zhao, Simin Guo, Ziqing Yang, Shifan Han, Dahua Lin, and Fei Tan. More data or better data? a critical analysis of data selection and synthesis for mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 618–629. Association for Computational Linguistics, 2025c. doi: 10.18653/v1/2025.emnlp-industry.43. URL https://aclanthology.org/2025.emnlp-industry.43/.

Kun Zhou, Beichen Zhang, Jiapeng Wang, Zhipeng Chen, Wayne Xin Zhao, Jing Sha, Zhichao Sheng, Shijin Wang, and Ji-Rong Wen. JiuZhang3.0: Efficiently improving mathematical reasoning by training small data synthesis models. *Advances in Neural Information Processing Systems*, 37:1854–1889, 2024.

Appendix guide. Appendix A collects the data, training, and evaluation settings, and Appendix B identifies external assets. Appendices C–E document the method's semantic vocabulary, actions, and algorithm. Appendix F gives additional results, Appendix G contains the prompts, and Appendix H illustrates generated problems. Appendix I discusses the scope and impacts of the work.

A Experimental and Reproduction Details

A.1 Data Construction and Comparison Protocol

The training comparison uses historical AMC/AIME problems from NuminaMath-CoT. As described in Section 4.1, exact, near-duplicate, and embedding-based comparisons against the six evaluation benchmarks leave 2640 seed problems. Each synthesis method produces 800 examples from this common pool. Training combines each synthetic set with the same 800 LIMO examples, giving a matched 1600-example corpus. The quality comparison in Section 4.3 uses seeds solved by Qwen3-Max; validity checks the problem and solution together, and effectiveness additionally requires that Qwen3-Max fail to solve the valid synthesized problem. Baseline model sources are identified in Appendix B.

A.2 GRPO Training

- Base model: `Qwen/Qwen3-4B-Instruct-2507`
- Initialization strategy: `base`
- Training GPUs: 8
- Number of epochs: 1
- Per-device train batch size: 1
- Gradient accumulation steps: 1
- Number of generations per prompt: 8
- Learning rate: 1×10^{-6}
- Learning rate scheduler: `cosine`
- Warmup ratio: 0.05
- Optimizer: `adamw_torch`
- KL coefficient β: 0.0005
- Maximum sequence length: 16384
- Maximum completion length: 16384
- Training sampling temperature: 0.6
- Training top-p: 0.95
- Logging steps: 1
- Save steps: 1
- Save total limit: 2
- Approximate throughput: 25 samples per GPU-hour.

A.3 Evaluation

- Evaluation backend: `vllm`
- Evaluation GPU: `1`
- Evaluation batch size: 8
- Maximum new tokens: 16384
- Evaluation temperature: 0.6
- Evaluation top-p: 0.95
- Tensor parallel size: 1
- Approximate throughput: 800 samples per GPU-hour.

A.4 Language and Embedding Models

- Data Synthesis and DAG Extraction: Gemini 3 Flash
- DAG Verification and Problem Generation: GPT-5.2
- Embedder: Qwen3-Embedding-0.6B(Zhang et al., 2025)
- Solver and Difficulty Rater: Qwen3-Max
- Difficulty Lowerbound: Qwen3-235B-A22B (Yang et al., 2025)

B Licenses and Terms of Use for Existing Assets

We use public datasets, pre-trained models, and open-source libraries, citing their creators at first use and summarizing licenses and terms below. Unless stated otherwise, these assets are used only for research and are not redistributed or repackaged.

Table 6: Existing assets used in this work. We cite the original creators and follow the stated licenses and terms of use for each asset.

Asset	Type	Version / Source	License	Use in this work
LIMO (Ye et al., 2025)	Dataset	GAIR/LIMO on Hugging Face; original 817-sample training set; accessed on May 5, 2026. https://huggingface.co/datasets/GAIR/LIMO	Apache License 2.0	Human-curated base data for GRPO training.
NuminaMath-CoT (LI et al., 2024)	Dataset	AI-MO/NuminaMath-CoT on Hugging Face; accessed on May 5, 2026. https://huggingface.co/datasets/AI-MO/NuminaMath-CoT	Apache License 2.0	Seed problems for mathematical data synthesis.
AIME24 (Zhang & Math-AI, 2024)	Dataset	evalscope/aime24 on ModelScope; accessed on May 5, 2026. https://www.modelscope.cn/datasets/evalscope/aime24	Apache License 2.0	Benchmark evaluation.
AIME25 (Zhang & Math-AI, 2025)	Dataset	evalscope/aime25 on ModelScope; accessed on May 5, 2026. https://www.modelscope.cn/datasets/evalscope/aime25	Apache License 2.0	Benchmark evaluation.
AIME26 (Zhang & Math-AI, 2026)	Dataset	evalscope/aime26 on ModelScope; accessed on May 5, 2026. https://www.modelscope.cn/datasets/evalscope/aime26	Apache License 2.0	Benchmark evaluation.
AMC22–24	Dataset	evalscope/amc_22-24 on ModelScope; accessed on May 5, 2026. https://www.modelscope.cn/datasets/evalscope/amc_22-24	Apache License 2.0	Benchmark evaluation.
Qwen3-4B-Instruct-2507 (Yang et al., 2025)	Pre-trained language model	Qwen/Qwen3-4B-Instruct-2507 on Hugging Face; accessed on May 5, 2026. https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507	Apache License 2.0	Base model for our GRPO training.
Qwen3-14B (Yang et al., 2025)	Pre-trained language model	Qwen/Qwen3-14B on Hugging Face; accessed on May 5, 2026. https://huggingface.co/Qwen/Qwen3-14B	Apache License 2.0	Used to reproduce the DESIGNER baseline.

Continued on next page

Table 6 (continued)

Asset	Type	Version / Source	License	Use in this work
Qwen2.5-Math-72B-Instruct (Yang et al., 2024)	Pre-trained language model	Qwen/Qwen2.5-Math-72B-Instruct on Hugging Face; accessed on May 5, 2026. https://huggingface.co/Qwen/Qwen2.5-Math-72B-Instruct	Qwen License Agreement	Used to reproduce the PromptCoT baseline.
PromptCoT Problem Generation Model (Zhao et al., 2025b)	Pre-trained problem generation model	zhaoxlpku/PromptCoT-Problem-Generation-Model on ModelScope; accessed on May 5, 2026. https://www.modelscope.cn/models/zhaoxlpku/PromptCoT-Problem-Generation-Model/summary	MIT License	Used for the PromptCoT baseline's problem generation pipeline.
Qwen3-Embedding-0.6B (Zhang et al., 2025)	Pre-trained embedding model	Qwen/Qwen3-Embedding-0.6B on Hugging Face; accessed on May 5, 2026. https://huggingface.co/Qwen/Qwen3-Embedding-0.6B	Apache License 2.0	Node embeddings for compatibility estimation.
Qwen3-235B-A22B (Yang et al., 2025)	Pre-trained language model	Qwen/Qwen3-235B-A22B on Hugging Face; accessed on May 5, 2026. https://huggingface.co/Qwen/Qwen3-235B-A22B	Apache License 2.0	Solver defining the lower difficulty bound for seed selection.

Baseline implementations and checkpoints. We reproduce DESIGNER with Qwen3-14B and PromptCoT with Qwen2.5-Math-72B-Instruct and the released PromptCoT Problem Generation Model, citing both original papers. These assets appear in Table 6. For baselines cited only for reported results, without using their code, checkpoints, datasets, or scripts, no additional asset license applies. Any released GRPO-trained checkpoint will credit Qwen3-4B-Instruct-2507 and include the license and notices required by its base model license.

C Semantic Features for Certain Domains

C.1 Algebra

Algebraic semantic features describe mathematical objects, from constants to piecewise functions; properties such as monotonicity and periodicity; and relations expressed through equalities, inequalities, and mappings. They support the formal representation and manipulation of algebraic expressions and equations.

- **Subject**: Constant, Variable, Monomial, Polynomial, RationalExpression, RadicalExpression, ExponentialFunc, LogarithmicFunc, AbsoluteValue, PiecewiseFunc, Sequence, Series, Matrix, Set, Interval
- **Attribute**: Value, Sign, Degree, Coefficient, Base, Exponent, Domain, Range, Periodicity, Parity, Monotonicity, Asymptote, Discriminant, Roots
- **Representation**: Equality, InequalityStrict, InequalityWeak, Identity, Approximation, System, Mapping, Membership

C.2 Number Theory

Number-theoretic semantic features describe integers, number sets, and arithmetic functions, with properties such as primality, parity, and divisibility. Representations include modular arithmetic, congruences, and prime factorizations for expressing relationships among these objects.

- **Subject**: IntegerVariable, RealVariable, Constant, NumberSet, Sequence, ArithmeticFunction
- **Attribute**: Value, Parity, Primality, Sign, Cardinality, DigitSum, IntegralPart

- **Representation**: AlgebraicEquation, Inequality, ModularCongruence, DivisibilityRelation, PrimeFactorizationForm, BaseExpansion, CoprimalityRelation

C.3 COMBINATORICS

Combinatorial semantic features describe sets and graphs, with quantitative attributes such as cardinality and degree and structural properties such as connectivity and distance. Set relations, graph properties, and logical implications support the formulation of optimization and enumeration problems.

- **Subject**: Set, Element, Sequence, Matrix, Proposition, Graph, Vertex, Edge
- **Attribute**: Cardinality, Value, TruthValue, Degree, Weight, Distance, Parity, Connectivity
- **Representation**: Equality, Inequality, SubsetOf, ElementOf, DisjointFrom, IsomorphicTo, AdjacentTo, IncidentTo, Implies, EquivalentTo

D REFERENCE ACTION LIST

D.1 ALGEBRA

Algebraic actions simplify and restructure expressions, solve equations, and apply identities. They include combining like terms, distributing products, completing the square, and applying logarithmic and exponential properties.

CombineLikeTerms, Distribute, Factorize, Rationalize, CommonDenominator, Substitute, IsolateVariable, Eliminate, CompleteSquare, ApplyFormula, LogarithmProperties, ExponentRules, CrossMultiply

D.2 NUMBER THEORY

Number-theoretic actions include the Euclidean algorithm for greatest common divisors, prime factorization, and Diophantine equation solving. Modular reduction and base conversion support different number systems; case enumeration explores finite solution spaces.

PrimeFactorize, ComputeGCD, ComputeLCM, ModuloReduce, SolveDiophantine, EuclideanAlgorithm, BaseConversion, EnumerateCases

D.3 COMBINATORICS

Combinatorial actions cover set operations (union, intersection, difference), counting operations (permutations, combinations, factorials), and logical operations (conjunction, disjunction, negation). Mapping transforms combinatorial structures for counting and optimization.

Union, Intersection, Difference, Complement, CartesianProduct, Permute, Choose, Factorial, Negate, Conjoin, Disjoin, Map

E PSEUDOCODE

Input : Seed problems $\mathcal{P}_{seed}$, semantic library $\mathcal{L}_{domain}$, corpus synthesis budget, and composition-size and length budgets
Output : New Problem Set $\mathcal{P}_{new}$

// **Stage 1: DAG Construction**
$\mathcal{G} \leftarrow \emptyset$
foreach $p \in \mathcal{P}_{seed}$ **do**
  $domain \leftarrow$ LLM_Identify_Domain(p)
  $G \leftarrow$ LLM_Extract_DAG$(p, \mathcal{L}_{domain})$ // $G = (V, A)$
  foreach $v \in V$ **do**
    $tags_v \leftarrow$ Extract_Semantic_Tags$(v, \mathcal{L}_{domain})$
    $v \leftarrow (v, tags_v)$
  end
  // Validate acyclicity, semantic tags, and step correctness
  if *Validate_DAG*(G) **then**
    $\mathcal{G} \leftarrow \mathcal{G} \cup \{G\}$
  end
end
// **Stage 2: Edge-Aware Compatibility Estimation**
$\mathcal{B} \leftarrow \mathcal{C}(\mathcal{G})$ // Node content, semantic tags, and incoming/outgoing contexts
// **Stage 3: Iterative Composition and Rendering**
$\mathcal{P}_{new} \leftarrow \emptyset$
while *corpus synthesis budget not reached and* $\mathcal{B} \neq \emptyset$ **do**
  // Select two distinct compatible DAGs within the composition budgets
  $(G_a, G_b, v_a, v_b) \leftarrow$ SelectFeasibleInitialBridge$(\mathcal{B})$
  if *no feasible initial tuple exists* **then**
    break
  end
  $j \leftarrow 2$
  $G_2^\star \leftarrow \mathcal{S}(G_a, G_b, v_a, v_b)$
  $\widehat{G}_2^\star \leftarrow \mathrm{Prop}(G_2^\star; v_a)$
  $\mathcal{U} \leftarrow \{G_a, G_b\}$ // Source DAGs already used in this composite
  while *composition-size and length budgets permit further extension* **do**
    // Check current edge-aware compatibility, source quality, and remaining budgets
    $\mathcal{F} \leftarrow$ FeasibleExtensions$(\widehat{G}_j^\star, \mathcal{G} \setminus \mathcal{U})$
    if $\mathcal{F} = \emptyset$ **then**
      break
    end
    // Consider marginal gain and future extensibility
    $(G_{j+1}, u_j, v_{j+1}) \leftarrow$ SelectExtension$(\mathcal{F})$
    if *the selected candidate offers insufficient marginal synthesis gain* **then**
      break
    end
    $G_{j+1}^\star \leftarrow \mathcal{S}(\widehat{G}_j^\star, G_{j+1}, u_j, v_{j+1})$
    $\widehat{G}_{j+1}^\star \leftarrow \mathrm{Prop}(G_{j+1}^\star; u_j)$
    $\mathcal{U} \leftarrow \mathcal{U} \cup \{G_{j+1}\}$
    $j \leftarrow j + 1$
  end
  $T \leftarrow j$
  // Render after stopping and apply the final quality check
  $p_{gen} \leftarrow \mathcal{R}(\widehat{G}_T^\star)$
  if *LLM_Quality_Score*$(p_{gen}) > \epsilon$ **then**
    $\mathcal{P}_{new} \leftarrow \mathcal{P}_{new} \cup \{p_{gen}\}$
  end
end
return $\mathcal{P}_{new}$

Algorithm 1: LEGO-MATH pipeline: DAG construction, compatibility estimation, and iterative composition.

F Additional Experimental Results

This appendix reports the run-to-run variability of the training results in the main paper. Experimental settings are provided in Appendix A, and the synthesis prompts are listed in Appendix G.

F.1 Run-to-Run Variability of Training Results

We report sample standard deviations across $N=8$ independent training runs that differ only in random seed. Training data, model checkpoint, hyperparameters, and evaluation protocol remain fixed. Each run uses the main experiments' decoding configuration on every benchmark; *Avg.* aggregates per-problem accuracies into a problem-weighted average. Lower standard deviations indicate more stable gains across seeds.

Table 7 reports the primary Qwen results, and Table 8 reports the corresponding results for Phi and Llama. On the primary backbone, LEGO-MATH attains both the highest mean accuracy and the smallest or near-smallest standard deviation across the six benchmarks, indicating that its improvements are consistent across seeds rather than driven by favorable runs.

Table 7: **Accuracy and sample standard deviations across 8 runs.** Each cell reports mean accuracy (%) over 8 seeds above the sample standard deviation in percentage points. **Bold** marks the best result per column; underline marks the second best (excluding our method). *Avg.* is problem-weighted across all 224 problems in six benchmarks. *# Train* is the training set size; "800+800" denotes LIMO plus 800 problems synthesized by the corresponding method.

Setting	# Train	AIME24	AIME25	AIME26	AMC22	AMC23	AMC24	Avg.
Baselines								
Qwen3-4B-Instruct-2507	0	51.67 ±7.35	35.83 ±3.45	37.08 ±4.52	78.49 ±4.26	81.52 ±3.85	69.72 ±3.93	62.50 ±4.24
LIMO	800	52.50 ±6.11	36.67 ±5.04	44.17 ±6.61	78.49 ±4.77	80.16 ±1.39	72.50 ±3.74	63.95 ±4.16
Text-level synthesis								
SAND-Math	800+800	53.33 ±7.97	37.50 ±5.56	44.58 ±7.96	77.03 ±3.39	84.24 ±3.02	69.44 ±3.09	64.17 ±4.59
CoT-Self-Instruct	800+800	52.50 ±4.96	32.50 ±3.88	41.67 ±3.98	77.33 ±3.46	83.42 ±4.18	72.22 ±4.28	63.45 ±3.96
Intermediate-representation synthesis								
PromptCoT	800+800	50.00 ±8.73	<u>39.17</u> ±6.61	43.33 ±8.16	<u>79.94</u> ±3.27	80.98 ±5.03	73.89 ±3.71	64.56 ±5.37
DESIGNER	800+800	51.25 ±5.89	34.58 ±2.48	41.67 ±9.59	78.49 ±4.08	83.97 ±4.49	72.78 ±3.09	64.01 ±4.60
Our Method								
LEGO-MATH (w/o DAG)	800+800	<u>56.67</u> ±1.78	**43.75** ±1.18	<u>50.83</u> ±2.36	73.26 ±5.27	<u>85.33</u> ±1.01	<u>77.78</u> ±1.19	<u>67.47</u> ±1.27
LEGO-MATH (ours)	800+800	**58.75** ±3.05	**43.75** ±1.18	**52.08** ±3.05	**84.30** ±2.71	**86.68** ±1.39	**78.06** ±2.77	**70.37** ±2.17

Table 8: **Cross-family accuracy and sample standard deviations across 8 runs.** Each cell reports mean accuracy (%) on the first line and sample standard deviation in percentage points on the second line. The *Avg.* column is problem-weighted across the six benchmarks. Best mean result(s) within each backbone are shown in **bold**.

Setting	# Train	AIME24	AIME25	AIME26	AMC22	AMC23	AMC24	Avg.
Phi-4-mini-instruct								
Base Model	0	5.00 ±3.09	4.58 ±3.05	5.00 ±3.56	21.51 ±2.98	23.37 ±6.34	20.28 ±3.65	14.96 ±1.24
LIMO	800	6.67 ±3.56	**5.42** ±3.05	**5.42** ±2.48	23.84 ±4.08	25.27 ±4.02	21.94 ±3.45	16.52 ±1.29
LEGO-MATH	800+800	**8.33** ±1.78	**5.42** ±4.34	**5.42** ±1.73	**24.71** ±3.50	**29.08** ±5.06	**25.28** ±2.89	**18.36** ±0.97
Llama-3.2-3B-Instruct								
Base Model	0	4.58 ±3.96	0.00 ±0.00	0.42 ±1.18	9.59 ±2.30	15.22 ±4.19	8.06 ±3.74	7.25 ±0.98
LIMO	800	4.58 ±4.34	0.83 ±1.54	0.83 ±1.54	13.66 ±3.61	**16.58** ±1.99	10.28 ±3.55	8.93 ±1.26
LEGO-MATH	800+800	**5.00** ±3.09	**1.25** ±2.48	**2.92** ±2.14	**15.70** ±6.55	**16.58** ±3.84	**10.83** ±4.51	**9.82** ±2.00

G PROMPTS

This section reproduces all LEGO-MATH prompts verbatim; {{ }} marks placeholders for runtime substitution. The figures cover problem checking, solving, judging, metadata and difficulty assessment, DAG extraction and refinement, bridge checking, and problem composition. Longer prompts appear in consecutive labeled parts. Shared variables, such as difficulty levels and topic tags, use consistent names.

【 TEMPLATE: PROBLEM CHECK 】

<Role>
You are a meticulous math problem analyst. Your task is to verify the clarity and completeness of the given math problem.
</Role>

<Instructions>
Just examine the problem statement for:
1. Clarity: Is the problem clearly stated without ambiguity?
2. Completeness: Is all necessary information provided to solve the problem?
</Instructions>

<Output Format>
Output ONLY a JSON object with the following keys:
{{
  "is_valid": true/false,
  "reason": "Brief explanation for the validity assessment."
}}
</Output Format>

Problem:
{problem}

Figure 7: Problem-checking prompt.

【 TEMPLATE: SOLVER 】

<Role>
You are a genius math solver. Reason step by step to solve the problem.
</Role>

Problem:
{problem}

Please put your final answer within \boxed{{}} if possible.

Figure 8: Solver prompt.

【 TEMPLATE: JUDGE 】

<Role>
You are a strict Math Judge. Your task is to zero in the problem and just evaluate the student's answer against the ground truth answer for the given math problem.
Output ONLY a JSON object with the following keys:
{{
- "status": can be "correct" (the answer and reasoning are both correct), "wrong" (the answer or the reasoning is incorrect), "error" (the student's answer is not even a valid attempt to answer the question).
- "reason": brief explanation for the judgement.
}}
</Role>

Problem:
{problem}
Ground Truth:
{gt}
Student Answer:
{student_ans}

Figure 9: Answer-judging prompt.

【 TEMPLATE: META EXTRACT 】

<Role>
You are a Math Knowlaction Expert.
</Role>

<Task>
Your task is listed as follow:
1. Analyze the problem type (select from: Multiple Choice, Fill in the Blank, True or False, Short Answer, Calculation, Numerical Proof, Proof, Other).
2. Analyze the question's domain and subdomain (select from: {domain}).
3. Analyze the concepts (e.g., inequality, probability).
4. Analyze the theorems used (e.g., Cauchy's Inequality).
5. Analyze the methods used.
6. Analyze the required abilities.
</Task>

<Output Format>
Please respond ONLY with a JSON string in the following format:
{{
  "problem_type": "String",
  "domain": ["String"],
  "subdomain": ["String"],
  "concepts": ["String"],
  "theorems": ["String"],
  "methods": ["String"],
  "required_abilities": ["String"]
}}
</Output Format>

Problem: {problem}
Solution: {solution}

Figure 10: Metadata-extraction prompt.

【 TEMPLATE: DIFFICULTY RATE 】

<ROLE>
You are an Expert Math Problem Difficulty Assessor.
</ROLE>

<TASK>
Analyze the provided math problem and solution to assign a difficulty score based on the provided reference materials.
</TASK>

<Difficulty Reference>
<Difficulty Level Descriptions (1.0 - 10.0 Scale)>
* 1.0 - 2.0 (Elementary Competition): Problems testing basic arithmetic, intuition, and direct application of elementary concepts (fractions, basic geometry, simple logic). Typical of **AMC 8** or early **AMC 10**. No complex algebra required; reachable by talented middle school students.
* 3.0 - 4.0 (Junior Varsity / Standard High School): Problems requiring standard high school algebra (quadratics, systems of equations), basic number theory (divisibility, primes), or plane geometry properties. requires multi-step reasoning but standard techniques. Typical of mid/late **AMC 10** or early **AMC 12**.
* 5.0 - 6.0 (Varsity / Invitational Prep): Problems requiring creative application of advanced high school topics (logarithms, complex numbers, recursion, modular arithmetic). "AIME Qualifier" level. Often requires combining two distinct mathematical areas. Typical of **AMC 12** (hard) or **AIME** (Questions 1-5).
* 7.0 - 8.0 (National Olympiad / Proof-Based): Rigorous proof-based problems. Requires deep knowledge of Olympiad theorems (e.g., Stewart's Theorem, Euler's Totient, Cauchy-Schwarz). The path to the solution is not visible immediately and requires constructing a logical argument. Typical of late **AIME** or **USA(J)MO**.
* 9.0 - 10.0 (World Class / IMO): Extremely difficult problems requiring novel insights, construction of new mathematical objects, or mastery of obscure/advanced theorems. Solutions are often lengthy or require a "genius" spark (e.g., Vieta Jumping, combinatorial nullstellensatz). Typical of **USAMO** (hardest) or **IMO**.
</Difficulty Level Descriptions (1.0 - 10.0 Scale)>

Figure 11: Difficulty-rating prompt: role and rating scale (part 1 of 3).

<Example Problems by Difficulty Level>

<Difficulty 1.0>
Concept: Basic Geometry / Perimeter
Problem: A square has an area of 64 square centimeters. If a smaller square with a perimeter of 12 centimeters is cut out from one corner, what is the perimeter of the remaining shape?
Solution: The side of the large square is 8 cm. The side of the small square is 3 cm. When a corner square is removed, the two new edges created exactly replace the two edges removed from the original perimeter. Thus, the perimeter remains $8 \times 4 = 32$ cm.
Analysis: Requires basic geometric intuition and understanding that "removing area" doesn't always "remove perimeter." No advanced formulas needed.
</Difficulty 1.0>

<Difficulty 3.5>
Concept: Algebra / Vieta's Formulas
Problem: Let r_1 and r_2 be the roots of the quadratic equation $x^2 - 7x + 11 = 0$. Find the value of $\frac{1}{r_1 + 1} + \frac{1}{r_2 + 1}$.
Solution: Let $y = x + 1$, so $x = y - 1$. Substitute into the equation: $(y-1)^2 - 7(y-1) + 11 = 0 \Rightarrow y^2 - 9y + 19 = 0$. The roots of this new equation are $y_1 = r_1+1$ and $y_2 = r_2+1$. We want the sum of the reciprocals: $\frac{1}{y_1} + \frac{1}{y_2} = \frac{y_1 + y_2}{y_1 y_2}$. By Vieta's, $y_1 + y_2 = 9$ and $y_1 y_2 = 19$. Result: $9/19$.
Analysis: Requires multi-step algebraic manipulation and knowledge of root relationships beyond just "plugging into the quadratic formula."
</Difficulty 3.5>

<Difficulty 5.5>
Concept: Combinatorics / Stars and Bars with Constraints
Problem: How many ways can 10 identical apples be distributed among 3 children such that each child receives at least 1 apple, but no child receives more than 5 apples?
Solution: First, give 1 apple to each child (3 total). We have 7 apples left to distribute to 3 children with the constraint $x_i \le 4$. Total ways without upper bound: $\binom{7+3-1}{3-1} = \binom{9}{2} = 36$.
Using Principle of Inclusion-Exclusion (PIE): Subtract cases where at least one child has > 4 (i.e., ≥ 5).
If child A has ≥ 5, we distribute $7-5=2$ apples: $\binom{2+3-1}{2} = 6$ ways. There are 3 children, so $3 \times 6 = 18$.
Intersection of two children having ≥ 5 is impossible ($5+5 > 7$).
 $36 - 18 = 18$.
Analysis: Requires combining a standard counting technique (Stars and Bars) with a secondary logic layer (PIE).
</Difficulty 5.5>

<Difficulty 7.5>
Concept: Geometry / Power of a Point
Problem: Let ABC be a triangle with circumcircle Γ. Let D and E be points on sides AB and AC respectively, such that $DB = BC = CE$. Let the circumcircle of $ \triangle ADE$ intersect Γ at A and Q. Prove that the incenter of $\triangle ABC$ lies on the line AQ.
Solution: This involves showing that AQ is the radical axis of the two circles. By using the property of the "Isogonal Conjugate" or showing that Q is the center of a spiral similarity, one can demonstrate that AQ bisects $\angle BAC$. The proof requires constructing power-of-a-point relations and recognizing the properties of the "Miquel Point" of a configuration.
Analysis: The solution is not algorithmic. It requires identifying an auxiliary point and constructing a rigorous geometric argument typical of USAMO-level geometry.
</Difficulty 7.5>

<Difficulty 10.0>
Concept: Combinatorics / Graph Theory (Ramsey-type)
Problem: (IMO Shortlist) In a mathematical competition, some competitors are friends. Friendship is mutual. Any group of 4 competitors contains a pair of friends. Prove that the competitors can be separated into two rooms such that in one room, every competitor has at most 7 friends, and in the other, every competitor has at most 10 friends. (Note: Original constants may vary by year).
Solution: This requires a sophisticated application of the "Extremal Principle." One might consider a partition that minimizes a specific weighted sum of "bad" edges (friendships within the same room). By showing that any "local minimum" of this cost function satisfies the problem's conditions, the existence of the partition is proved.
Analysis: Requires "moving" elements between sets to reach an optimal state—a classic high-level Olympiad strategy that involves no standard computation, only abstract logical construction.
</Difficulty 10.0>

</Example Problems by Difficulty Level>

</difficulty_reference>

Figure 12: Difficulty-rating prompt: reference examples (part 2 of 3).

```
<INSTRUCTIONS>
1. Analyze: Carefully read the provided 'Math Problem' and its 'Solution'. Identify the core mathematical concepts,
required techniques, and the complexity of the argument. Note any particularly clever steps, non-obvious insights, or
reliance on advanced theorems.
2. Compare: Compare the analyzed problem to the 'Example Problems by Difficulty Level' and the 'Difficulty Level
Descriptions' provided in the reference materials. Consider where it fits in terms of typical competition level (AMC
8/10/12, AIME, USA(J)MO, IMO) and the type of thinking required.
3. Score: Assign a difficulty score between **1.0 and 10.0**, using increments of **0.5** (e.g., 3.0, 3.5, 4.0). The score must
be consistent with the provided reference scale.
4. Summarize: Write a brief paragraph summarizing the problem's core topic and mathematical area(s). Enclose this
summary within '<S>' and '</S>' tags.
5. Assign Score: Place the difficulty score assigned in step 3.
</INSTRUCTIONS>

<OUTPUT FORMAT>
ONLY a JSON object with the following keys:
{{
   "difficulty_score": [float] (The assigned score between 1.0 and 10.0, in increments of 0.5),
   "summary": [str] (Your brief paragraph summarizing the problem),
   "justification": [str] (Your paragraph justifying the score based on analysis and comparison to references)
}}
</OUTPUT FORMAT>

Problem: {problem}
Solution: {solution}
```

Figure 13: Difficulty-rating prompt: instructions and output format (part 3 of 3).

【 TEMPLATE: DAG EXTRACT 】

<Role>
You are a Senior Mathematical Knowledge Graph Engineer.
Your core task is to deeply parse a mathematical problem (stem, solution, and answer) into a semantically precise and logically complete Directed Acyclic Graph (DAG).
</Role>

<Core Objective>
Transform the linear solution text into a structured graph where:
1. Nodes (States) are atomic information snapshots (formulas, values, facts).
2. Actions (Edges) are the logical operators that transform Input Nodes into Output Nodes.
</Core Objective>

<Part 1: Node Definitions (States)>
Guideline: Atomicity & Purity
* Atomic: Each node must contain exactly one distinct mathematical statement. Split compound steps.
* Pure: Content must be raw math/logic. **REMOVE** cohesive text ("Therefore", "Since"), equation labels ("Eq.1"), and placeholders.

Classify nodes into these strict types:

A. Condition Node (Given Facts)
* `Value Constraint`: Specific numbers/limits (e.g., $a=5$, $x > 0$).
* `Structural Constraint`: Shapes, definitions (e.g., "Triangle ABC is isosceles").

B. Hypothesis Node (Assumptions)
* `Unknown Variable Assumption`: Introduction of unknowns (e.g., "Let x be...").
* `Case Assumption`: Branch initiation (e.g., "Case 1: $x \ge 0$").
* `Contradiction Assumption`: Setup for proof by contradiction.
* `Inductive Hypothesis`: The $n=k$ assumption.

C. Knowledge Node (External Principles)
* `Theorem/Formula`: Named theorems or formulas cited (e.g., "Law of Cosines").
* Note: This is the rule itself, not the calculation result.*

D. Goal Node (Direction)
* `Target`: The final question asked.
* `Sub-Goal`: Intermediate objective (e.g., "Find discriminant Δ").

E. Derived Node (New Information)
* `Intermediate Value`: Calculated results.
* `Transformed Expression`: Simplification, factorization, or substitution results.
* `Logical Inference`: Qualitative deductions (e.g., "Function f is monotonic").

F. Answer Node (Terminal)
* `Final Value`: The numerical answer.
* `Proof Conclusion`: The Q.E.D. statement.
</Part 1: Node Definitions (States)>

<Part 2: Action Definitions (Edges)>
Guideline: Completeness & Accuracy
* Self-Contained Logic: An Action is valid ONLY if its `inputs` provide *all* necessary information to generate the `outputs`. If an Action is isolated, the derivation $Inputs
ightarrow Outputs$ must still hold mathematically.
* Topology Rules:
  * Condition/Knowledge Nodes are typically Sources (Inputs).
  * Derived/Answer Nodes are Sinks (Outputs).
  * Do NOT set a Condition Node as an output of a calculation.

Figure 14: DAG-extraction prompt (part 1 of 2).

```
Action Structure:
For each logical step, identify:
1. Type: Strictly select from `{reference_actions}`.
2. Description: A concise, verb-based summary (e.g., "Substitute $x$ and simplify").
3. Inputs: List of Node IDs required (Conditions, Previous Derived Nodes, or Knowledge).
4. Outputs: List of Node IDs generated by this action.
</Part 2: Action Definitions (Edges)>

<Reference Data>
Allowed Action Types:
{reference_actions}
</Reference Data>

<Output Format>
Output ONLY a JSON object.
Ensure STRICT DAG topology (NO cycles, NO floating nodes).
{{
  "nodes": [
    {{
      "id": "N1",
      "type": "Condition Node",
      "subtype": "Value Constraint",
      "content": "f(x) = x^2 + 2x"
    }},
    ...
  ],
  "actions": [
    {{
      "id": "A1",
      "type": "Differentiate",
      "description": "Differentiate f(x) with respect to x",
      "inputs": ["N1", "N2"],
      "outputs": ["N3"]
    }},
    ...
  ]
}}
</Output Format>

Problem: {problem}
Solution: {solution}
```

Figure 15: DAG-extraction prompt (part 2 of 2).

```
【 TEMPLATE: DAG VERIFY 】
-----------------------------------------------------

<Role>
You are a Math Logic Validator.
</Role>

<Instructions>
You may be given a derivation step extracted from a mathematical problem's solution process, represented as a transformation from [Input States] to [Output States] via an [Action].
Your task is to determine whether this step is logically valid or not based on the provided information, and provide a concise reasoning for your judgment.
</Instructions>

<Output Format>
Output ONLY JSON: {{
   "reason": "...",
   "is_valid": true/false
}}
</Output Format>


[Input States]:
{input_states}

[Action]:
{action}

[Output States]:
{output_states}
```

Figure 16: DAG-verification prompt.

```
【 TEMPLATE: STATE ENRICH 】
-----------------------------------------------------
<Role>
You are a Math Semantic Analyst. Your goal is to tag the mathematical state with precise semantic features suitable for the problem domain.
</Role>

<Domain Context>
The problem belongs to the following domains. Choose the most relevant ONE from these domains to analyze the State Content, unless you find them irrelevant.
{semantic_feature_context}
</Domain Context>

<Instructions>
Analyze the specific semantic role and complexity of the given State Content and respond in JSON: {{
   "Subject": "String",
   "Attribute": "String",
   "Representation": "String"
}}
</Instructions>

State Content: {state_content}
```

Figure 17: State-enrichment prompt.

```
【 TEMPLATE: DAG REFINE 】
------------------------------------------------------

<Role>
You are a Senior Mathematical Expert and a Graph Engineer.
You will be provided with:
1. The original Problem and Solution.
2. A Logic DAG in JSON format generated previously, which reprentsents the reasoning steps from the solution with some errors.
3. An Error Report indicating some of the structural or semantic flaws in the Logic DAG.
</Role>

<Task>
Your goal is to fix all the errors and regenerate a completely new, valid, and self-contained Logic DAG JSON.
Do not just output the diff. You must output the full corrected graph.
</Task>

<DAG Definition>
Transform the linear solution text into a structured graph where:
1.  Nodes (States) are atomic information snapshots (formulas, values, facts).
2.  Actions (Edges) are the logical operators that transform Input Nodes into Output Nodes.
</DAG Definition>

<Part 1: Node Definitions (States)>
Guideline: Atomicity & Purity
* Atomic: Each node must contain exactly one distinct mathematical statement. Split compound steps.
* Pure: Content must be raw math/logic. **REMOVE** cohesive text ("Therefore", "Since"), equation labels ("Eq.1"), and placeholders.

Classify nodes into these strict types:

A. Condition Node (Given Facts)
* `Value Constraint`: Specific numbers/limits (e.g., $a=5$, $x > 0$).
* `Structural Constraint`: Shapes, definitions (e.g., "Triangle ABC is isosceles").

B. Hypothesis Node (Assumptions)
* `Unknown Variable Assumption`: Introduction of unknowns (e.g., "Let $x$ be...").
* `Case Assumption`: Branch initiation (e.g., "Case 1: $x \ge 0$").
* `Contradiction Assumption`: Setup for proof by contradiction.
* `Inductive Hypothesis`: The $n=k$ assumption.

C. Knowledge Node (External Principles)
* `Theorem/Formula`: Named theorems or formulas cited (e.g., "Law of Cosines").
* Note: This is the rule itself, not the calculation result.*

D. Goal Node (Direction)
* `Target`: The final question asked.
* `Sub-Goal`: Intermediate objective (e.g., "Find discriminant $\Delta$").

E. Derived Node (New Information)
* `Intermediate Value`: Calculated results.
* `Transformed Expression`: Simplification, factorization, or substitution results.
* `Logical Inference`: Qualitative deductions (e.g., "Function $f$ is monotonic").

F. Answer Node (Terminal)
* `Final Value`: The numerical answer.
* `Proof Conclusion`: The Q.E.D. statement.
</Part 1: Node Definitions (States)>

<Part 2: Action Definitions (Edges)>
Guideline: Completeness & Accuracy
* Self-Contained Logic: An Action is valid ONLY if its `inputs` provide *all* necessary information to generate the `outputs`. If an Action is
isolated, the derivation $Inputs
ightarrow Outputs$ must still hold mathematically.
* Topology Rules:
   * Condition/Knowledge Nodes are typically Sources (Inputs).
   * Derived/Answer Nodes are Sinks (Outputs).
   * Do NOT set a Condition Node as an output of a calculation.
```

Figure 18: DAG-refinement prompt (part 1 of 2).

Action Structure:
For each logical step, identify:
1. Type: Strictly select from {reference_actions}.
2. Description: A concise, verb-based summary (e.g., "Substitute x and simplify").
3. Inputs: List of Node IDs required (Conditions, Previous Derived Nodes, or Knowledge).
4. Outputs: List of Node IDs generated by this action.
</Part 2: Action Definitions (Edges)>

<Refinement Guidelines>
1. Address Critical Errors: Specifically target the issues listed in the Error Report (e.g., isolated nodes, disconnected logic, wrong calculations).
2. Preserve Correct Logic: Keep the parts of the DAG that are mathematically sound.
3. Strict Topology: Ensure the graph is fully connected from Conditions to Answers. No floating nodes.
</Refinement Guidelines>

<Output Format>
Output ONLY the complete JSON object.
{{
  "nodes": [...],
  "actions": [...]
}}
</Output Format>

[Problem]:
{problem}

[Original Solution]:
{solution}

[Logic DAG Need Refinement]:
{previous_dag_str}

[Error Report]:
{error_report}

Please fix the DAG based on the report.

Figure 19: DAG-refinement prompt (part 2 of 2).

```
【 TEMPLATE: BRIDGE CHECK 】
----------------------------------------------------

<Role>
You are a Math Logic Gatekeeper.
Your task is to validate if an Upstream Provider CAN logically connect to a Downstream Receiver based on a Semantic Cluster.
</Role>

<Classification Choices>
1. INVALID:
  - Direct conflict (e.g., Triangle vs Square).
  - Logical impossibility.
  - Completely irrelevant.

2. VALID:
  - The connection makes semantic sense.
  - We can potentially adapt the Downstream input to match Upstream.
</Classification Choices>

<Task>
1. Analyze the semantic compatibility.
2. If VALID, provide a `relation_hint`. This is a mathematical or logical instruction on how to transform the Upstream Value to become the
Downstream Input.
  - Default (Equality): "Input = Upstream_Value"
  - Shift: "Input = Upstream_Value - 6"
  - Extraction: "Input = Coefficient of t in Upstream_Value"
</Task>

<Output Format>
Return purely a JSON object:
{{
   "relation_hint": "Formula or instruction (e.g., 'Target = Source - 6')" if VALID, or brief explanation if INVALID
   "decision": "INVALID" or "VALID",
}}
</Output Format>


[Upstream Context]:
- Output: {upstream_node_content}

[Downstream Context]:
- Input: {downstream_node_content}

[Proposed Bridge]:
Connect using Cluster Semantic: "{cluster_concept}"

[Task]:
Analyze and Classify.
```

Figure 20: Bridge-checking prompt.

```
【 TEMPLATE: VAR RENAME 】
------------------------------------------------------

<Role>
You are a Mathematical Variable Scope Manager.
Your task is to rename ALL variables in a specific Instance of a computation graph to ensure namespace isolation.
</Role>

<Input>
A sequence of DAG items (Nodes and Actions) sorted in topological execution order.
</Input>

<Instructions>
1. Identify mathematical variables (e.g., 'x', 'y', 'radius', 'theta', 'S_ABC').
2. Append the provided suffix `_{inst_suffix}` to every variable name, unless it already ends with that suffix.
3. STRICTLY preserve numbers, operators, standard functions (sin, cos, log), and logic text.
4. Maintain the consistency of the variable flow. If 'x' becomes 'x_A' in the Condition, it MUST be 'x_A' in the subsequent Actions and Results.
</Instructions>

<Example>
Suffix: "P1"
Input: {{"id": "N1", "content": "x = 5"}}, {{"id": "A1", "desc": "Calculate x^2"}}, {{"id": "N2", "content": "y = 25"}}
Output: {{"N1": "x_P1 = 5", "A1": "Calculate (x_P1)^2", "N2": "y_P1 = 25"}}
</Example>

<Output Format>
Return a JSON Map where keys are Item IDs and values are the renamed 'content' (for nodes) or 'description' (for actions).
</Output Format>

<Input>
[Instance Suffix]: {inst_suffix}
[Execution Flow (Topological Order)]:
{items_json}
</Input>
```

Figure 21: Variable-renaming prompt.

```
【 TEMPLATE: ADAPT CONDITIONS SINGLE 】
------------------------------------------------------

<Role>
You are a Math Logic Engine.
We are trying to FORCE a connection between two Math DAGs (Hard Bridge).
</Role>

<Goal>
Rewrite the `New Inlet Value` using `Upstream Value` and `Relation Hint`.
- Example: If Upstream Value is "a+b=5" and Relation Hint is "Input = Upstream_Value" and the Downstream Inlet is "r=3", then New Inlet
Value is "r=5".
</Goal>

<Input Format>
1. Upstream Value: The finalized value from the previous part.
2. Relation Hint: How Upstream affects Downstream.
3. Downstream Inlet (Original): The node receiving the value.
4. Downstream Conditions (Original): The constraints defining the downstream problem.
</Input Format>

<Output>
JSON Object:
{{
   "success": true/false,
   "new_inlet_value": "The calculated string for the inlet node"
}}
</Output>

<Inputs>
[Upstream Value]: {upstream_val}
[Relation Hint]: {relation_hint}
[Downstream Inlet Original]: {d_inlet_original}
</Inputs>
```

Figure 22: Condition-adaptation prompt.

【 TEMPLATE: DISSOLVE INLET 】

<Role>
You are a Mathematical Editor.
Your task is to rewrite the content of an "Inlet Node" to hide its specific value and instead reference its source (Upstream).
</Role>

<Instructions>
{dissolve_inlet_instructions}
If the HINT is not sufficient or wrong, use general mathematical language to express the relationship and ensure clarity.
</Instructions>

<Output Format>
Return JUST the new text string. No JSON.
</Output Format>

<Dissolve Inlet Node Content>
[Current Node Content]: {current_val}
[Upstream Source Description]: {source_desc}
[Relation Hint]: {hint}
</Dissolve Inlet Node Content>

Figure 23: Inlet-dissolution prompt.

【 TEMPLATE: VALUE PROPAGATE 】

<Role>
You are a precise Mathematical Calculation Engine.
Your task is to update the content of Output Nodes based on Input Nodes, strictly following the LOGIC of the designated Operation (Action). If the action includes original inputs, you can just take them as a reference and use the provided Input Node values for calculation.
</Role>

<Rules>
1. PRESERVE VARIABLES: If an input is "x=10", output "y=20". Do not change variable names provided in the Output Node Context.
2. BATCH CALCULATION: Calculate values for ALL Output Nodes listed.
3. FORMAT: Output valid JSON only. The keys must be the exact Node IDs provided in the [Context].
4. LaTeX: Keep LaTeX formatting if present. Escape backslashes properly in JSON (e.g., "\sqrt" -> "\\sqrt").
</Rules>

<Output Format>
Response Format Example:
{{
  "node_uuid_1": "x = 5",
  "node_uuid_2": "y = 25"
}}
</Output Format>

<Input Data>
[Action Logic]:
Type: {action_type}
Description: {action_desc}

[Input Nodes - Original Values]:
{parents_original}

[Output Nodes - Original Context]:
{child_original}

[Input Nodes - New Values]:
{parents_current}
</Input Data>

<Instruction>
Calculate the new content for the Output Nodes based on the New Inputs.
</Instruction>

Figure 24: Value-propagation prompt.

【 TEMPLATE: PROBLEM COMPLETE 】

<Role>
You are a Math Problem Architect.
I have a "Reasoning Graph" (DAG) representing the complete logical flow of a math problem.
You construct coherent, self-contained math problems based on rigorous logical graphs.
Your task is to write the **Problem Statement** and **Solution Outline** text based on this graph.
[Instructions]:
1. Identify Givens: Look for "Condition Nodes" (usually input nodes with no parents) or "Knowledge Nodes". These are your knowns.
2. Identify Goal: Look for the "Answer Node" (sink node) or "Goal Node". This is what the student needs to find. If it is a conclusion or proof, find a way to phrase it as a number or value to compute if possible.
3. Hide the Middle: The "Derived Nodes" and "Actions" are the solution path. Do NOT reveal them explicitly in the problem text. Use them to understand the context so you can describe the scenario naturally.
4. Through the Bridge: There may exist "Bridge Actions" that connect the former derived nodes to the latter conditions. Use these to create a smooth narrative flow and do not expose the calculation steps and the original numerical values.
5. Drafting: Write a single, polished problem statement.
6. Solution Outline: Provide a brief outline of the solution steps BASED ON THE GRAPH structure, and highlight the final ANSWER.
</Role>

<Output Format>
Output the problem text and accompanying solution outline FROM THE GRAPH in the following format:
</Output Format>

<Problem Statement>
<The full problem text here>
</Problem Statement>

<Solution Outline>
<Brief solution steps and the answer here>
</Solution Outline>

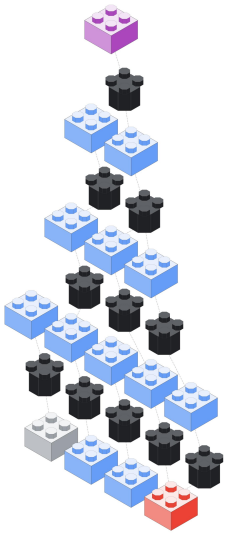
<Graph Structure>
Nodes (Information States):
{all_nodes_text}

Actions (Logical Steps):
{all_actions_text}
</Graph Structure>

Figure 25: Problem-completion prompt.

H CASE STUDIES

Figures 26–28 present three representative examples illustrating the multi-domain reasoning and structural complexity of the generated problems.



P 17

Domain： Number Theory， Algebra

Problem:

Let x and y be two-digit positive integers with a mean of 60. Let the maximum possible value of the ratio $\frac{x}{y}$ be written in simplest form as $\frac{a}{b}$, where a and b are coprime positive integers.

Define the integer M by the equation $M = 3(a + b) + 3$. Compute the smallest positive integer n for which the decimal representation of n^n ends in the digits of M (i.e., $n^n \equiv M \pmod{10^k}$ where k is the number of digits in M).

Solution:

To solve this problem, we first determine the values of x and y. We start with the given constraints: $x, y \in \mathbb{Z} \cap [10, 99]$ and the mean condition $\frac{x+y}{2} = 60$.

We **Calculate the sum of x and y from their mean**, which gives us the equation $x + y = 120$.

Our goal is to find $max(\frac{x}{y})$. To achieve this, we **Determine x and y to maximize the ratio x/y given constraints**. Since $x + y = 120$, maximizing x/y requires maximizing x and minimizing y. With $x \leq 99$ and $y \geq 10$, we find $x = 99, y = 21$.

Next, we **Substitute the values of x and y into the ratio**, resulting in $\frac{x}{y} = \frac{99}{21}$. We then **Simplify the fraction to its coprime form** to get $\frac{a}{b} = \frac{33}{7}$. From this, we **Extract coprime integers a and b**, yielding $a = 33, b = 7$.

Now we calculate M using the formula $M = 3(a + b) + 3$. We **Calculate M using the values of a and b**, which gives $M = 123$.

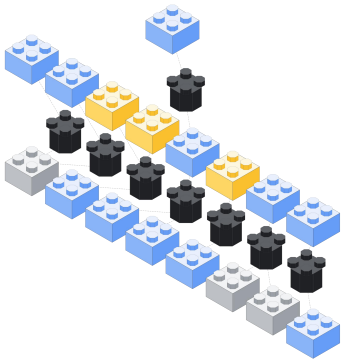
We then **Count the number of digits in M** and find $k = 3$. With this, we define our target congruence. We **Set up the congruence equation for n**: we need to find n such that $n^n \equiv 123 \pmod{1000}$.

To solve this, we analyze the congruence modulo 8 and 125 separately. First, we **Reduce the congruence modulo 8** to get $n^n \equiv 3 \pmod{8}$. We **Solve the congruence for n modulo 8**, determining that $n \equiv 3 \pmod{8}$.

Simultaneously, we **Reduce the congruence modulo 125**, leading to $n^n \equiv 123 \pmod{125}$. We **Solve the congruence for n modulo 125 using lifting and totient properties** and find $n \equiv 117 \pmod{125}$.

Finally, we **Solve the system of congruences using the Chinese Remainder Theorem**. Combining $n \equiv 3 \pmod{8}$ and $n \equiv 117 \pmod{125}$ yields $n \equiv 867 \pmod{1000}$. We **Identify the smallest positive integer n satisfying the congruence**, concluding that the answer is 867.

Figure 26: **A LEGO-MATH case study illustrating multi-step reasoning across Number Theory and Algebra.** The problem first uses the mean condition on two-digit positive integers x and y to derive $x + y = 120$, then maximizes the ratio $\frac{x}{y}$ under the given constraints to obtain $\frac{99}{21} = \frac{33}{7}$. From this, the value $M = 3(a + b) + 3 = 123$ is computed. The remaining task is transformed into a modular arithmetic problem: finding the smallest positive integer n such that $n^n \equiv 123 \pmod{1000}$. The reasoning trace is decomposed into discrete steps, with different colors representing distinct types of reasoning nodes, including algebraic calculation, modular reduction, Chinese Remainder Theorem synthesis, and final answer identification. The final result is $n = 867$. In the accompanying LEGO visualization, different colors denote distinct types of reasoning nodes, such as calculation, logical deduction, and final synthesis, highlighting the structural trajectory of the solution.



P 18

Domain： Number Theory， Combinatorics

Problem:

Part 1:Consider the set of all positive integers a of the form $a = 2^n 3^m$, where n and m are non-negative integers, such that a^6 is not a divisor of 6^a . Let S be the sum of all such integers a .Part 2:Jacob plays a round of golf consisting of N holes, where $N = \frac{S}{6} + 2$. On the k -th hole (for $k = 1, 2, \ldots, N$), Jacob achieves a score of exactly k (i.e., 1 on the first hole, 2 on the second, etc.). Each hole is assigned a par value of 3, 4, or 5. For each hole, Jacob calculates a difference value defined as the score on that hole minus its par. Given that the set of these N differences can be arranged to form a sequence of N consecutive integers, calculate the total number of distinct ways the pars could have been assigned to the holes.

Solution:

Part 1: Calculating S

We start with the form of integer a : $a = 2^n 3^m$, subject to $n, m \in \mathbb{Z}_{\geq 0}$. The divisibility condition is given as $a^6 \mid 6^a$.

We **Express a^6 in terms of n and m** to get $a^6 = 2^{6n} 3^{6m}$. Similarly, we **Express 6^a in terms of n and m** yielding $6^a = 2^a 3^a$.

We then **Convert divisibility condition to exponent inequality**. The condition $a^6 \mid 6^a$ implies $6n > a \vee 6m > a$. We **Consider the two cases of the OR condition**:

1. $6n > 2^n 3^m$
2. $6m > 2^n 3^m$

For the first case, we **Solve for a in Case 1**, finding $a \in \{2, 4, 8, 16\}$.

For the second case, we **Solve for a in Case 2**, finding $a \in \{3, 9\}$.

We **Combine unique values of a** to get the set $a \in \{2, 3, 4, 8, 9, 16\}$. We then **Calculate the sum S**, resulting in $S = 42$.

Part 2: Combinatorial Analysis

We determine N using the formula $N = \frac{S}{6} + 2$. We **Calculate the number of holes N** to find $N = 9$.

We **Define the set of holes for the calculated N**: let the hole index be $k \in \{1, 2, \ldots, N\}$. The par values are constrained by $p_k \in \{3, 4, 5\}$ and the difference is $d_k = k - p_k$.

We **Determine the possible values for the difference d_k** and find $d_k \in \{k - 5, k - 4, k - 3\}$. The problem states that $\{d_1, d_2, \ldots, d_N\}$ is a set of N consecutive integers .

Using these constraints, we **Identify possible ranges for the sequence of differences**. We deduce that **The range of $\{d_k\}$ must be** $\{-4, \ldots, 4\}, \{-3, \ldots, 5\},$ or $\{-2, \ldots, 6\}$.

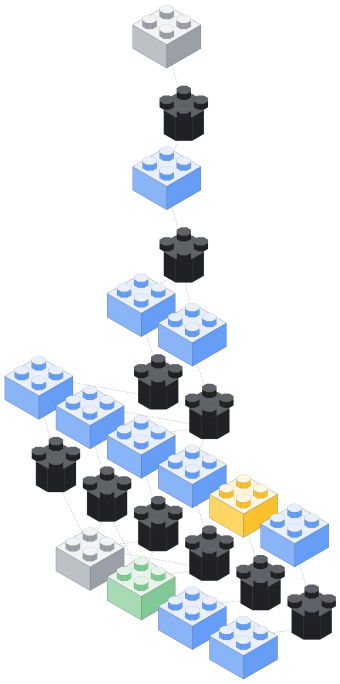
We analyze each case:

1. We **Count assignments for the range starting at -4**. We find **The number of valid par assignments for range $\{-4, \ldots, 4\}$ is 1**.
2. We **Count assignments for the range starting at -2**. We find **The number of valid par assignments for range $\{-2, \ldots, 6\}$ is 1**.
3. For the middle range, we **Analyze the range starting at -3 as a bijection problem**. This implies **For range** $\{-3, \ldots, 5\}$, d_k is a bijection from $\{1, \ldots, 9\}$ to $\{-3, \ldots, 5\}$. We **Map the difference bijection to a restricted permutation problem**: $\sigma(k) = d_k + 4$ is a permutation of $\{1, \ldots, 9\}$ such that $\sigma(k) \in \{k - 1, k, k + 1\}$.

Using the theorem that **The number of permutations σ of** $\{1, \ldots, L\}$ such that $|\sigma(k) - k| \leq 1$ is F_{L+1} , we **Apply the Fibonacci formula to count permutations**. This gives **The number of valid par assignments for range** $\{-3, \ldots, 5\}$ is $F_{10} = 55$.

Finally, we **Sum the number of ways from all three cases** to get 57 .

Figure 27: **A LEGO-MATH case study illustrating multi-step reasoning across Number Theory and Combinatorics.** The problem spans the domains of Number Theory and Combinatorics. The reasoning trace is decomposed into discrete states corresponding to the numbered solution steps: Part 1 derives the sum $S = 42$ through exponent inequalities, while Part 2 maps the combinatorial par-assignment to a restricted permutation problem. In the accompanying visualization, different colors represent distinct types of reasoning nodes (e.g., calculation, logical branching, and final synthesis) to highlight the model's structural trajectory.



P 24

Domain： Combinatorics, Number Theory

Problem:

Five towns are connected by a system of roads such that there is exactly one road connecting each pair of towns. Let K be the number of ways to assign a one-way direction to every road so that it is possible to travel from any town to any other town (possibly passing through intermediate towns).

Now, consider the set $S = \{1, 2, \ldots, 2K\}$. Let $\mathcal{A}$ be the collection of all subsets $A \subset S$ such that $|A| = K$ and no element in A divides another (i.e., for all distinct $x, y \in A$, $x \nmid y$). For any such set A, let m_A denote the smallest element in A. Determine the minimum possible value of m_A.

Solution:

Part 1: Determining K

The system describes **Five towns connected by roads forming a complete graph** K_5. We **Calculate the number of edges in a complete graph with 5 vertices**, finding **Total number of edges** $E = \binom{5}{2} = 10$. We then **Calculate total possible orientations of the edges**, which gives **Total number of orientations of the edges** = $2^{10} = 1024$.

Our objective is to **Find K, the number of strongly connected tournaments on 5 vertices.** We use the theorem: **A tournament on 5 vertices is strongly connected if and only if it has no source (in-degree 0) or sink (out-degree 0).**

We use the Principle of Inclusion-Exclusion:

- We **Calculate the number of tournaments with at least one source: Number of orientations with a source** = $\binom{5}{1} \times 2^6 = 320$.
- We **Calculate the number of tournaments with at least one sink: Number of orientations with a sink** = $\binom{5}{1} \times 2^6 = 320$.
- We **Calculate the number of tournaments with both a source and a sink: Number of orientations with both a source and a sink** = $5 \times 4 \times 2^3 = 160$.

We **Apply the Principle of Inclusion-Exclusion to find total invalid orientations**, finding **Total number of non-strongly connected orientations** = $320 + 320 - 160 = 480$.

We **Subtract invalid orientations from total orientations to find K**, resulting in $K = 1024 - 480 = 544$.

Part 2: Analyzing Subset A

We define $S = \{1, 2, \ldots, 2K\} = \{1, 2, \ldots, 1088\}$. The subset A satisfies $A \subset S, |A| = 544$ and $\forall x, y \in A, x \nmid y$.

We assume the property that **The set S can be partitioned into 544 disjoint chains $C_u = \{u \cdot 2^k \in S\}$ for each odd number $u \leq 1088$.** We **Deduce that A must pick exactly one element from each chain to reach size K**, establishing that **A must contain exactly one element from each of the 544 chains C_u.**

To minimize the smallest element, we focus on chains related to powers of 3. We **Select specific elements from chains with odd parts that are powers of 3. Let $x_i = 3^i \cdot 2^{e_i} \in A$ be the elements chosen from chains with odd parts $3^0, 3^1, \ldots, 3^6$.**

We **Determine the constraint on exponents required by the anti-chain property**. This implies The anti-chain condition $x_i \mid x_j$ for $i < j$ implies the sequence of exponents is strictly decreasing: $e_0 > e_1 > e_2 > e_3 > e_4 > e_5 > e_6 \geq 0$.

We **Find the minimum value of the largest exponent in a strictly decreasing sequence of 7 non-negative integers** and determine **The minimum possible value for e_0 is 6.**

We then **Calculate the minimum value of the element corresponding to the odd part 1**. This gives **The element from the chain C_1 is $x_0 = 1 \cdot 2^{e_0} \geq 2^6 = 64$.**

We aim to **Determine the minimum possible value of $m_A = min(A)$**. Based on the above, we **Conclude that the minimum possible value of the smallest element in A is 64**, giving the final answer **64**.

Figure 28: **A LEGO-MATH case study illustrating multi-step reasoning across combinatorics and number theory.** The problem first asks for the number K of strongly connected orientations of the complete graph K_5. Part 1 counts these orientations by applying inclusion–exclusion to all 2^{10} tournaments on five vertices, obtaining $K = 544$. Part 2 studies the divisibility anti-chain condition on $S = \{1, 2, \ldots, 1088\}$, reducing it to a selection problem over chains of the form $C_u = \{u \cdot 2^k\}$ indexed by odd integers u, and concludes that the minimum possible value of m_A is 64. In the accompanying LEGO visualization, different colors denote distinct types of reasoning nodes, such as calculation, logical deduction, and final synthesis, highlighting the structural trajectory of the solution.

I LIMITATIONS AND BROADER IMPACTS

I.1 LIMITATIONS

Reliability Boundary. LEGO-MATH combines solution-DAG extraction, semantic labeling, compatible-node matching, downstream value propagation, and final verification to improve sample validity and training gains. This reliability rests on structural constraints and empirical verification rather than formal proof. DAG extraction, semantic adaptation, and natural-language rendering still rely on model judgment; implicit conditions, variable constraints, branching assumptions, or phrasing artifacts may produce invalid samples. Integrating CAS, SMT solvers, theorem provers, or multi-model verifiers could strengthen verification.

Scope of DAG Recomposition. We study iterative multi-DAG composition by stacking an upstream fragment of the current composite with a downstream fragment of a new DAG through compatible bridge nodes. Each extension undergoes constrained value propagation; feasibility, marginal synthesis gain, and synthesis budgets determine composition depth. This keeps compatibility checking, value propagation, and quality analysis tractable while providing effective training signals. The implementation excludes parallel fusion, conditional fusion, and arbitrary graph editing. These extensions require finer-grained semantic adaptation, stronger constraint management, and more sophisticated rendering control.

Scope of Evaluation. LEGO-MATH improves synthetic sample quality and GRPO post-training results in our setting. Effectiveness measures difficulty relative to a reference solver, not absolute mathematical difficulty; DAG height is an empirical proxy for reasoning depth, not a complete difficulty metric. Experiments focus on algebra, number theory, and combinatorics, with one primary base model and training configuration. Additional solvers, model scales, training algorithms, domains, and larger evaluation sets are needed to characterize generalization and applicability.

I.2 BROADER IMPACTS

Expanding access to structured mathematical training data. High-quality mathematical reasoning data improves large language models, but manually constructing competition-level problems and complete solutions is costly. LEGO-MATH recombines existing solution DAGs while preserving traceable dependencies, enabling controlled data expansion for post-training. Such data may provide denser, more targeted reasoning signals for small and mid-scale models. This approach can reduce data construction costs and make synthetic reasoning data easier to analyze.

Appropriate use of synthesized problems. LEGO-MATH data are intended for model training, augmentation, and research, and require review before educational or assessment use. Despite DAG constraints, semantic adaptation, and verification, residual errors include missing implicit conditions, ambiguous statements, and problem–solution mismatches. Filtering, sample inspection, and statistical evaluation can mitigate noise in training. In teaching or assessment, errors may mislead learners or compromise fairness, so additional human review, external symbolic verification, or multi-model cross-checking should be applied.

Transparency in synthetic-data pipelines. Solution DAGs make synthesis more interpretable and auditable by exposing how samples are composed, which states are bridged, and where errors arise. This supports systematic comparison of data quality and downstream training utility. However, over-filtering, selective reporting, and opaque evaluation protocols can still affect synthetic-data pipelines. Releasing auditable samples, failure cases, filtering rules, and evaluation protocols would support fair comparison and reproducibility.