

Training LLMs for Divide-and-Conquer Reasoning

Anonymous ACL submission

Abstract

Large language models (LLMs) have demonstrated strong reasoning capabilities through step-by-step chain-of-thought (CoT) reasoning. Nevertheless, at the limits of model capability, CoT often proves insufficient, and its strictly sequential nature constrains test-time scalability. A potential alternative is divide-and-conquer (DAC) reasoning, which decomposes a complex problem into subproblems to facilitate more effective exploration of the solution. Although promising, our analysis reveals a fundamental misalignment between general-purpose post-training and DAC-style inference, which limits the model’s capacity to fully leverage this potential. To bridge this gap and fully unlock LLMs’ reasoning capabilities on the most challenging tasks, we propose an end-to-end reinforcement learning (RL) framework to enhance their DAC-style reasoning capacity. At each step, the policy decomposes a problem into a group of subproblems, solves them sequentially, and addresses the original one conditioned on the subproblem solutions, with both decomposition and solution integrated into RL training. Under comparable training, our DAC-style framework endows the model with a higher performance ceiling and stronger test-time scalability, surpassing CoT by 8.6% in *Pass@1* and 6.3% in *Pass@32* on competition-level benchmarks. The code is available at the [provided link](#).

1 Introduction

Large language models (LLMs) have shown remarkable capabilities in complex reasoning tasks, with chain-of-thought (CoT) enabled by large-scale reinforcement learning (RL) during post-training. Models such as OpenAI O1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025) can effectively solve competition-level mathematical problems like AIME (MAA) through long CoT reasoning with extensive self-reflection. However, on more challenging tasks, including Interna-

tional Mathematical Olympiad and advanced theorem proving, standard CoT often exhibits limited effectiveness, motivating the development of more advanced reasoning paradigms for LLMs.

An intuitive and classical approach to solving complex problems is the **divide-and-conquer** (DAC) strategy, originally developed in computer programming (Cormen et al., 2022). For LLMs, early approaches such as Tree-of-Thought (Yao et al., 2023) and DeAR (Xue et al., 2024) adopt similar strategies by problem decomposition and employing structured reasoning to solve subproblems, thereby facilitating the final solution. Recent works such as Seed-Prover (Chen et al., 2025) and DeepSeek-Prover-V2 (Ren et al., 2025) also incorporate DAC-style problem-solving approaches to tackle frontier mathematical tasks, underscoring the promise of DAC-style test-time scalability in fully eliciting the reasoning capacity of LLMs.

Despite establishing a DAC reasoning framework, these approaches function only during inference and rely on complex prompt engineering, leaving the model misaligned between its general post-training and this specific reasoning style, as complex reasoning problems are typically trained using step-by-step CoT. This misalignment can constrain the model’s DAC-style reasoning even on simpler problems that could be solved with CoT, as illustrated in Figure 1. Accordingly, we evaluate a wide range of instruction-following and reasoning models on mathematical benchmarks using DAC and CoT-style inference, and observe that most models exhibit inferior performance under direct DAC reasoning compared to CoT, as shown in Figure 2. This gap indicates that fully unlocking DAC-style problem-solving capabilities in general LLMs necessitates dedicated training.

To address this issue, we propose an end-to-end RL framework for equipping LLMs with DAC-style reasoning capabilities for tackling complex problems. We adopt RL over supervised fine-

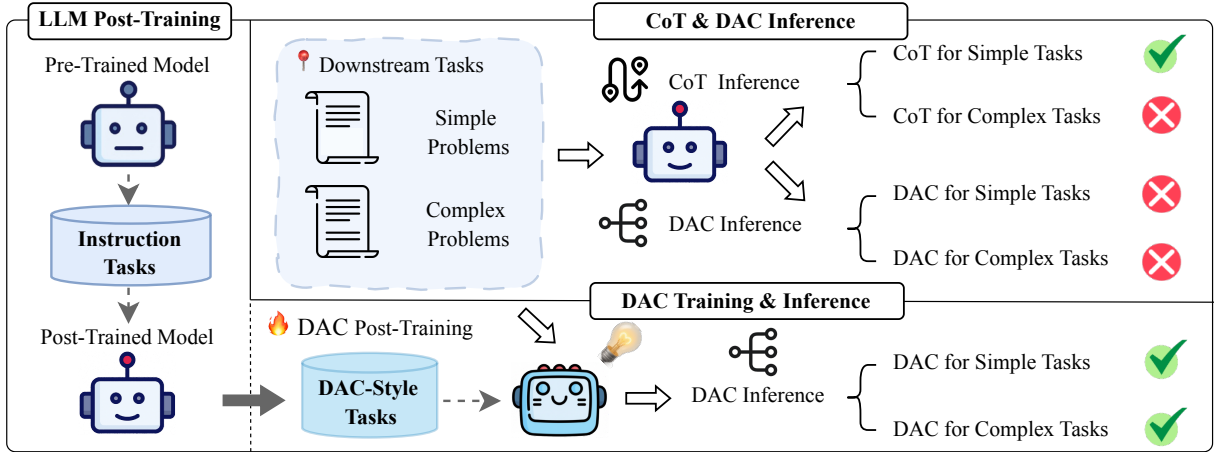


Figure 1: Overview of the LLM post-training pipeline, comparison of the DAC and CoT-style inference, and our proposed DAC post-training. Without dedicated training, DAC inference remains ineffective, whereas DAC-specific post-training makes this advanced reasoning paradigm robust across downstream tasks.

tuning because, when the training framework is applied to frontier models, stronger expert annotations are often unavailable or prohibitively expensive, whereas RL enables effective training through self-exploration. Specifically, at each RL iteration, the policy model decomposes every input problem into a set of subproblems. These are then aggregated with the original problem into a conquering prompt, which directs the model to solve the subproblems sequentially and then formulate the final answer. Both problem division and conquering are incorporated into unified policy training, with the corresponding optimization objectives defined in Sections 2.2 and 2.3, respectively. Furthermore, the evolution of distinct and improved subproblems over training epochs facilitates broader, non-repetitive exploration during conquering, which is crucial for keeping generation diversity and preventing entropy collapse (Liang et al., 2025).

To assess the effectiveness of our DAC-RL framework, we experiment with two models and evaluate the $Pass@1$ (averaged 32 times) and $Pass@32$ performance on challenging, widely used competition-level reasoning benchmarks, including AIME 24 & 25 (MAA), Beyond-AIME (ByteDance-Seed, 2025) and HMMT-25 (Balunović et al., 2025). Models trained under our framework consistently outperform standard CoT on the adopted benchmarks, achieving absolute margins of 8.6% and 6.3% on the $Pass@1$ and the $Pass@32$ metrics when using Qwen3-4B-Instruct-2507 as the initial policy, whereas CoT-style RL fails to improve performance over this post-trained model. Our analysis indicates that

DAC-style training is even beneficial for the policy’s CoT reasoning ability. Moreover, DAC-trained models exhibit more flexible and greater test-time scalability ceilings, along with more compact reasoning traces. We also experiment with cold-start initialization and evaluate its effectiveness in training models for DAC reasoning.

Our *contributions* are summarized as follows: (1) By comparing DAC and CoT at inference and integrating DAC into training, we reveal a critical misalignment between general-purpose post-training and DAC inference that constrains performance. (2) We introduce DAC-RL, a unified training framework that strengthens DAC reasoning via RL, mitigates the reasoning paradigm’s bottlenecks, and raises the performance ceiling of LLMs on challenging problems. (3) We perform extensive experiments on post-trained LLMs and evaluate the trained models on challenging benchmarks with detailed analysis, demonstrating the superiority of the proposed DAC-RL framework.

2 Method

2.1 Preliminaries

Task Formalization In this section, we formalize our DAC-style reasoning and CoT reasoning (Wei et al., 2022). In standard CoT reasoning, given an input problem x , the policy model π_θ directly generates a step-by-step reasoning trajectory y , from which the final answer can be extracted as $a = \text{Extract}(y)$. For DAC-style reasoning, given an input x , the model first performs a *division* step, generating a response y_d consisting of a set of n_g subproblems derived from x ,

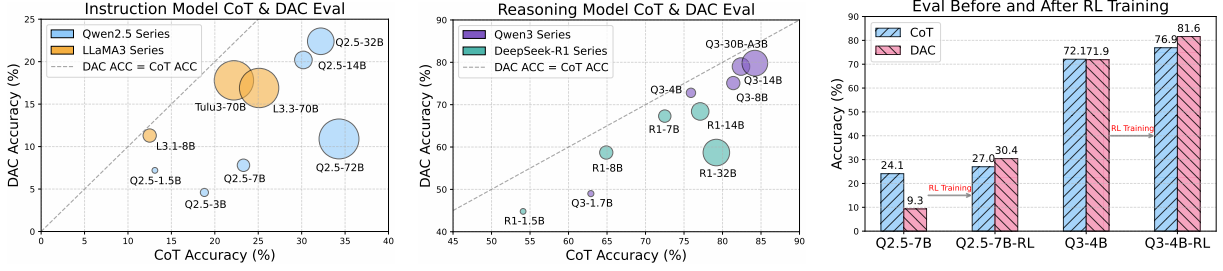


Figure 2: We evaluate the CoT and DAC $Pass@32$ performance on four competition-level benchmarks (Table 1) for both general instruction and reasoning post-trained models. The right panel presents $Pass@32$ performance for the Qwen2.5-7B-Instruct and Qwen3-4B-Instruct-2507 models before and after task-specific RL training.

denoted as $\mathcal{P} = \{p_i\}_{i=1}^{n_g} \sim \pi_\theta(\mathcal{P} | x)$. Next, in the *conquering* step, the generated subproblems \mathcal{P} together with x are concatenated to construct a *conquering* prompt, which guides the model to sequentially generate solutions to the subproblems as $\mathcal{S} = \{s_i\}_{i=1}^n \sim \pi_\theta(\mathcal{S} | x, \mathcal{P})$, and subsequently solve the original problem conditioned on \mathcal{S} . The complete *conquering* response is denoted as y_c .

The *overview* of our DAC training and inference pipeline is illustrated in Figure 3. At each RL training step, the policy model processes every problem in the batch by performing two tasks: (1) *Division*: Dividing the problem into a set of subproblems (green), and (2) *Conquering*: Sequentially solving the subproblems and then the original problem (blue). The learning objective is to maximize the expected rewards of both the *division* and *conquering* responses, defined as:

$$\mathcal{J}(\theta) = \mathbb{E}_{y_d, y_c \sim \pi_\theta} [\mathbf{R}(y_d) + \mathbf{R}(y_c)], \quad (1)$$

where $\mathbf{R}(y_d)$ and $\mathbf{R}(y_c)$ are the *division* and *conquering* rewards detailed in Section 2.2 and 2.3.

2.2 Subproblem Division

During each iteration, the policy is first prompted to decompose each problem x in the training set into G_d groups of subproblems $\{\mathcal{P}_g\}_{g=1}^{G_d}$, using the prompt in Figure 12. Notably, the policy is required to generate more than N_s subproblems for each input; without this constraint, it collapses to producing no useful subproblems in *division* response y_d as the training goes, and the *conquering* stage degrades to directly solving the original problem, as in standard CoT reasoning.

For training subproblem division, we adopt a combined reward scheme that integrates *format validity*, *quantity validity*, and its *helpfulness* in facilitating the solution of the original problem. Specifically, for *format validity*, we require that the subproblems generated by the policy in re-

sponse to the *division* prompt must be parsable using regular expressions. For *quantity validity*, since the policy is instructed to generate at least N_s subproblems, a negative reward is assigned when the number of extractable subproblems is fewer than N_s . For *helpfulness*, we evaluate each generated subproblem group \mathcal{P}_g using the accuracy of the solutions in the *conquering* stage $CA(\mathcal{P}_g)$ as an indicator of their helpfulness. Based on empirical analysis, we optimize a lower bound on subproblem helpfulness by requiring that each subproblem group \mathcal{P}_g yield at least one correct solution to the original problem, provided that the policy can generate a correct solution from any subproblem group in $\{P_i\}_{i=1}^{G_d}$. This encourages each subproblem group, when incorporated into the *conquering* prompt, to help guide the policy toward producing at least one correct solution for simple problems, while rewarding groups that enable correct solutions for challenging problems where others fail. In summary, the reward scheme for *division* generation y_d is derived as:

$$\mathbf{R}(y_d) = \begin{cases} 0, & \underbrace{|\mathcal{P}_g| < N_s}_{\text{quantity}} \vee \underbrace{\neg \text{Format}(y_d)}_{\text{format}} \\ 0, & \underbrace{CA(\mathcal{P}_g) = 0 \wedge CA(\{P_i\}_{i=1}^{G_d}) > 0}_{\text{helpfulness}} \\ 1, & \text{otherwise,} \end{cases} \quad (2)$$

where $|\mathcal{P}_g|$ denotes the number of subproblems, CA represents the *conquering* accuracy, and $\neg \text{Format}(y_d)$ indicates that y_d violates the required format, making \mathcal{P}_g non-extractable.

2.3 Subproblem and Original Conquering

Once the subproblem group \mathcal{P}_g is generated, it is combined with the original problem to form the *conquering* prompt (Figure 13), which guides the model to solve the subproblems sequentially and

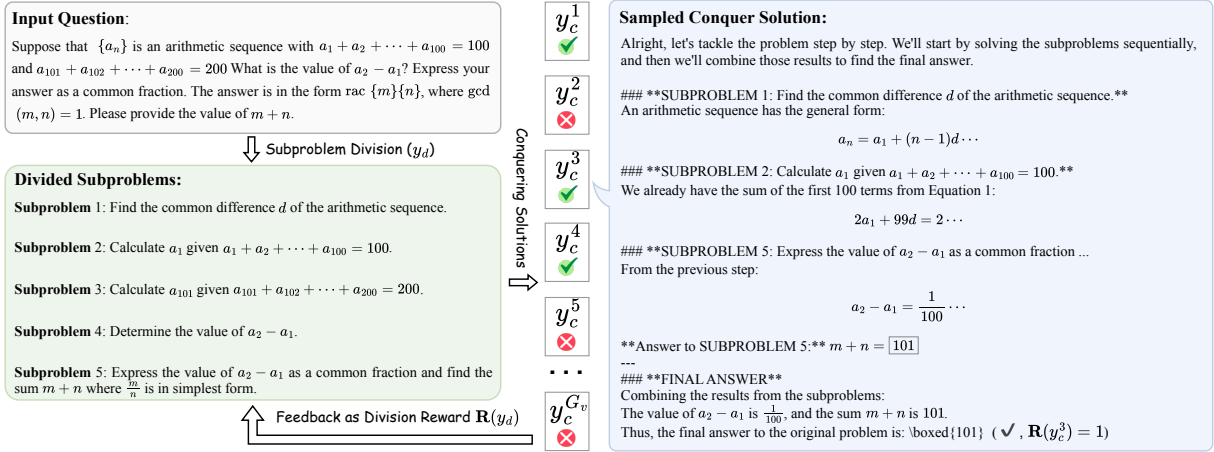


Figure 3: An overview of the DAC-style inference and reward assignments in training, illustrated with a case study. The policy decomposes the original problem into a group of subproblems, samples candidate *conquering* solutions in parallel, and leverages their correctness to compute *division* rewards for optimizing problem decomposition.

then tackle the original one. Since ground-truth answers for the subproblems are unavailable and therefore cannot serve as training signals, we instead rely solely on the correctness of the final answer extracted from y_c to the original problem as a surrogate reward for both subproblem solving and original problem solving. The underlying idea is that if the policy solves the subproblems correctly, their solutions can support answering the original problem, enabling the policy to solve it accurately. Conversely, incorrect solutions to the subproblems may lead to failure in solving the original problem. The theoretical validity of this reward scheme is established in Lemma 1. The final *conquering* reward is defined as follows:

$$\mathbf{R}(y_c) = \mathbf{1}\{\text{Extract}(y_c) = a\} \quad (3)$$

where a denotes the ground-truth answer to the original problem.

Lemma 1 (Final-answer reward positively associates with subproblem correctness). *Let $s_i \in \{0, 1\}$ indicate whether subproblem i is solved correctly and $\mathbf{s} = (s_1, \dots, s_m)$. Let $C \in \{0, 1\}$ indicate whether the original problem is solved correctly. If the division-conquer pipeline induces the causal direction $\mathbf{s} \rightarrow C$, we have:*

$$\text{Cov}_\theta(\mathbf{1}\{s_i = 1\}, \mathbf{1}\{C = 1\}) \geq 0, \quad (4)$$

where θ denotes the parameters of the policy π_θ . The proof is in Appendix B. Consequently, the scalar reward in Equation 3, which satisfies $\mathbb{E}_{y_c \sim \pi_\theta}[\mathbf{R}(y_c)] = P_\theta(C = 1)$, is a consistent surrogate signal for subproblem correctness: rewarding $C = 1$ preferentially upweights trajectories with more correct subproblems, and vice versa.

3 Experiments

3.1 Settings

Models and Datasets. We conduct experiments using two models, [Qwen2.5-7B-Instruct](#) and [Qwen3-4B-Instruct-2507](#), to evaluate the proposed DAC-RL framework. We use [DAPO-Math-17k](#) (Yu et al., 2025) as the training dataset. Following rStar2 (Shang et al., 2025), which suggests that rule-based verifiers often struggle with certain open-ended mathematical formats, we evaluate model performance only on benchmarks with integer answers to ensure precise assessment. Specifically, we evaluate models on four widely used competition-level mathematical reasoning benchmarks: AIME 2024, AIME 2025 (MAA), Beyond-AIME (ByteDance-Seed, 2025), and HMMT-25 (Balunović et al., 2025).

Implementation Details. For RL training, all experiments employ the GRPO (Shao et al., 2024) optimization strategy, extended with the Clip-Higher and token-level loss techniques (Yu et al., 2025). In each iteration, every input problem is divided into $G_d = 4$ subproblem groups containing varying numbers of subproblems, and each group, together with the original problem, is used to generate $G_c = 8$ *conquering* solutions. The minimum requirement of subproblems is set to $N_s = 3$. The training batch size is set to 256, the maximum rollout length to 8,192, and the sampling temperature to 1.0. For policy optimization, the Clip-Higher upper bound ε_h is fixed at 0.28, and the mini-batch size is 64. We conduct 400 training steps (nearly 6 epochs) for both models on the [DAPO-Math-17k](#) dataset, with an additional about 200

Algorithm 1 Divide-and-Conquer Reinforcement Learning for LLMs

Require: Training set \mathcal{D} ; initial policy π_θ ; division group size G_d ; conquering group size G_c ; training steps T

Ensure: Trained policy $\pi_{\theta'}$

```
1: Initialize experience buffer  $\mathcal{B} \leftarrow \emptyset$ 
2: for  $t = 1$  to  $T$  do
3:   Sample a mini-batch  $d \sim \mathcal{D}$ 
4:   for each problem–answer pair  $(x, a)$  in  $d$  do
5:     [Divide] Generate  $G_d$  subproblem groups  $\{\mathcal{P}_g\}_{g=1}^{G_d} \sim \pi_\theta(x)$ ,  $\mathcal{P}_g = \{p_{g,i}\}_{i=1}^{n_g}$ ,  $n_g$  varies per group.
6:     for each subproblem group  $\mathcal{P}_g$  do
7:       [Conquer] Generate  $G_c$  solution candidates  $\{y_{g,v}\}_{v=1}^{G_c} \sim \pi_\theta(\mathcal{P}_g; x)$ 
8:       Compute correctness rewards  $\{\mathbf{R}(y_{g,v})\}_{v=1}^{G_c}$  for conquering w.r.t. reference  $a$ 
9:       Store conquer tuples in buffer:  $\mathcal{B} \leftarrow \mathcal{B} \cup \{([x; \mathcal{P}_g], y_{g,v}, \mathbf{R}(y_{g,v}))\}_{v=1}^{G_c}$ 
10:    end for
11:    [Division Reward] Evaluate Format and Quantity validity  $\{f_g\}_{g=1}^{G_d}$ ,  $\{q_g\}_{g=1}^{G_d}$  for  $\{\mathcal{P}_g\}_{g=1}^{G_d}$ 
12:    Compute division rewards  $\{\mathbf{R}(\mathcal{P}_g)\}_{g=1}^{G_d}$  from  $\{\mathbf{R}(y_{g,v})\}_{v=1}^{G_c}$ ,  $\{q_g\}$ , and  $\{f_g\}$  via Eq. (2)
13:    Store division tuples in buffer:  $\mathcal{B} \leftarrow \mathcal{B} \cup \{(x, \mathcal{P}_g, \mathbf{R}(\mathcal{P}_g))\}_{g=1}^{G_d}$ 
14:  end for
15:  [Policy Update] Update policy  $\pi_\theta$  using  $\mathcal{B}$  according to Eq. (6)
16:  Clear buffer:  $\mathcal{B} \leftarrow \emptyset$ 
17: end for
18: return trained policy  $\pi_{\theta'}$ 
```

steps for the [Qwen3-4B-Instruct-2507](#) policy on a difficulty-filtered subset containing 3.7k problems that the initial policy solves with less than 50% accuracy across 16 responses, to better elicit its advanced reasoning capability.

For inference, we set the maximum token length to 16,384 unless otherwise specified, with the temperature kept the same as training (1.0) and the top- p value set to 0.7 following (Yu et al., 2025). We report the *Pass@1* metric as the average over 32 runs. For *Pass@32* evaluation, we employ an unbiased estimation method (Chen et al., 2021) to reduce the high variance from single evaluations.

3.2 Main Results

DAC-style Reasoning Shows a Higher Ceiling. We present the main results of our DAC-RL training and inference alongside the CoT baselines in Table 1. Compared with the standard CoT RL training, DAC-style training demonstrates substantially greater gains in *Pass@32* performance on competition-level benchmarks, achieving overall improvements of **3.4%** and **6.3%** for the [Qwen2.5-7B-Instruct](#) and [Qwen3-4B-Instruct-2507](#) models, respectively. Notably, these training improvements are achieved even when the initial policy’s DAC reasoning lags behind the CoT-style reasoning obtained through intensive post-training, particularly for the [Qwen2.5-7B-Instruct](#) model, which attains only 0.4% average DAC ac-

curacy on competition-level problems.

To characterize training effects across the two reasoning styles, we evaluate intermediate checkpoints in Figure 4. Notably, both models begin with lower DAC accuracy than CoT, particularly on the *Pass@1* metric. However, as training progresses, DAC performance steadily grows, eventually outperforming CoT and increasing at a faster pace. The steady improvements, together with the observation that CoT-style RL fails to elicit further gains for [Qwen3-4B-Instruct-2507](#), suggest that CoT reasoning is largely saturated in post-training for general instruction-tuned models. In contrast, DAC surpasses the CoT performance ceiling on the evaluated benchmarks, indicating a higher upper bound on reasoning and the necessity of dedicated DAC post-training to fully unlock it.

Deep DAC Training Further Enhances Reasoning. The motivation for DAC-style reasoning is to fully unleash the model’s potential to solve the most challenging problems at the edge of its capability. To this end, we conduct experiments of the *Deep DAC* training setting, training the [Qwen3-4B-Instruct-2507](#) model exclusively on the 3.7k most difficult subset described in Section 3.1 for ten epochs, with extended training and inference token budgets of 16,384 and 24,576, respectively. This setup equips the model with deeper mastery of DAC-style reasoning while enhancing test-time scalability, particularly for complex tasks. We fur-

Model	AIME 2024		AIME 2025		Beyond-AIME		HMMT 2025		Average	
	Pass@1	Pass@32	Pass@1	Pass@32	Pass@1	Pass@32	Pass@1	Pass@32	Pass@1	Pass@32
Qwen2.5-7B-Instruct										
Init-CoT	9.8	26.7	6.8	36.7	3.8	23.0	2.0	10.0	5.6	24.1
Init-DAC	0.5	13.3	0.2	6.7	0.7	10.0	0.2	6.7	0.4	9.2
RL-CoT	13.5	34.5	11.4	30.8	5.1	25.5	2.7	13.1	8.2	27.0
RL-DAC	15.5	39.1	15.5	34.2	7.0	27.4	4.8	20.8	10.4	30.4
Δ (RL)	+2.0	+4.6	+4.1	+3.4	+1.9	+1.9	+2.1	+7.7	+2.2	+3.4
Qwen3-4B-Instruct-2507										
Init-CoT	62.6	90.0	45.7	76.7	32.1	65.0	30.3	56.7	42.7	72.1
Init-DAC	59.6	90.0	43.2	73.3	29.6	61.0	28.2	63.3	40.2	71.9
RL-CoT	45.9	85.8	52.1	77.4	30.4	58.1	21.8	54.4	37.5	69.0
RL-DAC	63.9	87.7	54.2	78.8	34.6	67.9	31.9	66.6	46.1	75.3
Δ (RL)	+18.0	+1.9	+2.1	+1.4	+4.2	+9.8	+10.1	+12.2	+8.6	+6.3
Qwen3-4B-Instruct-2507 (Deep)										
RL-D-CoT	64.4	84.8	58.8	87.9	38.9	69.5	37.6	65.5	49.9	76.9
RL-D-DAC	66.3	91.6	61.5	87.6	38.8	70.7	38.7	76.4	51.3	81.6
Δ (RL)	+1.9	+6.8	+2.7	-0.3	-0.1	+1.2	+1.1	+10.9	+1.4	+4.7

Table 1: The results of the baselines and our DAC strategies are reported across six benchmarks using different models. The *Pass@1* metric is averaged over 32 runs to ensure a more stable and precise evaluation. **RL-D** refers to the *Deep DAC* setting with the baseline described in Section 3.2. Best results are highlighted in **bold**.

ther compare this configuration with CoT-style RL training under an identical budget of 32 rollouts per training problem, with results shown in the right panel of Figure 4 and the bottom section of Table 1. It is worth noting that simply increasing the rollout budget for CoT reasoning during RL training does not yield performance improvements, while DAC-style training delivers a **4.7%** improvement in *Pass@32* over the CoT baseline. This underscores the superior scalability of DAC-style reasoning during both training and inference.

4 Analysis

4.1 Integrating CoT- and DAC-RL Training

In this section, we experiment with integrating CoT- and DAC-RL training (*Mix-RL*), applying DAC-RL only to challenging problems while retaining CoT-style training for simpler ones, with experiments performed on [Qwen3-4B-Instruct-2507](#). Specifically, all batch problems are first answered in CoT-style rollouts, and those with accuracy below $t_{acc} = 25\%$ are replaced by DAC-style solutions, which are mixed into the experiences with the reward assignment in Section 2. We compare this Mix-RL training with both the full CoT-RL and DAC-RL strategies in Figure 5, from which we draw the following observations:

DAC Training Enhances CoT Reasoning. We find that substituting CoT training with DAC training on challenging problems can paradoxically enhance the policy’s CoT reasoning, even with re-

duced CoT training. As shown in Figure 5, the *Left* and *Middle* plots illustrate the intermediate performance under CoT-style inference for the CoT-RL and Mix-RL training across all benchmarks and on the AIME benchmarks, respectively. We find that further CoT-style RL training yields no additional gains, as the policy has already been extensively post-trained with CoT reasoning. However, under the Mix-RL setup, incorporating DAC-style training on challenging problems significantly enhances the model’s CoT performance—by over **10%** across all benchmarks. These improvements highlight the effectiveness and robustness of DAC-style training in model’s reasoning capability.

Mix-RL can Activate DAC Reasoning. The DAC-style reasoning performance of Mix-RL is shown in Figure 5 (*Right*). Notably, applying DAC-style training only to complex problems can also equip the model with this reasoning capability, consistently surpassing its CoT counterpart on the benchmarks, which reinforces the motivation for adopting a DAC strategy for complex tasks. The inferior performance of Mix-RL relative to full DAC suggests that incorporating a broader range of problems into DAC training is generally beneficial for this advanced reasoning paradigm.

4.2 Test-time Scalability and Configurations

We evaluate test-time scalability and investigate the optimal configuration of DAC-style reasoning by measuring *Pass@k* under a fixed rollout budget of $k = 1024$, while varying the number of sub-

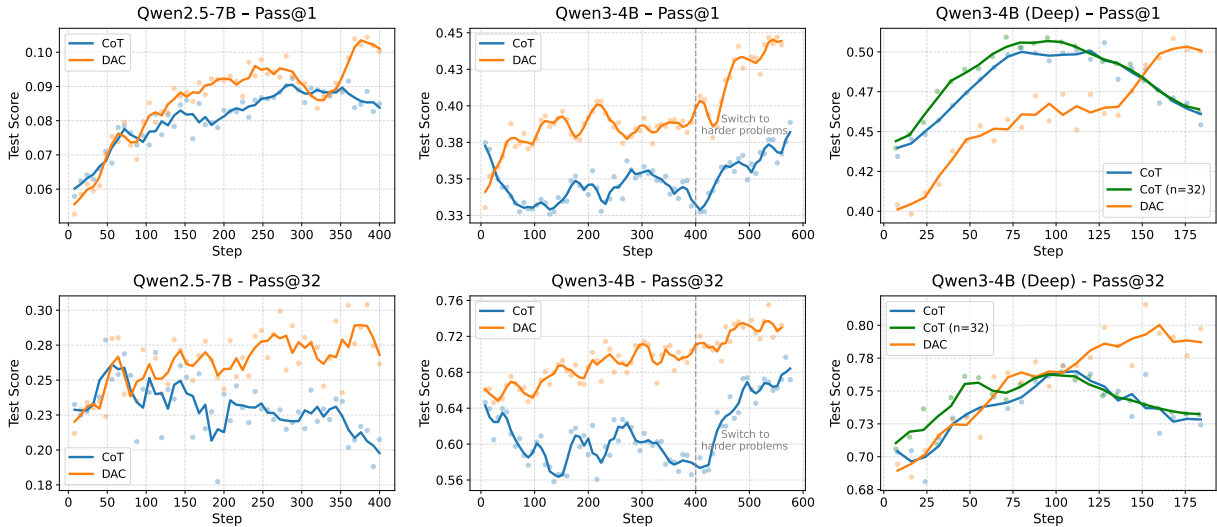


Figure 4: Intermediate evaluations across all four benchmarks during DAC-RL and CoT-RL training. **Middle:** For experiments using *Qwen3-4B-Instruct-2507*, the training set is updated to the difficult subset after the 400th iteration, as detailed in Section 3.1. **Right:** Results of our *Deep DAC* training experiments, additionally including a comparison with the CoT-RL baseline using 32 rollouts in training.

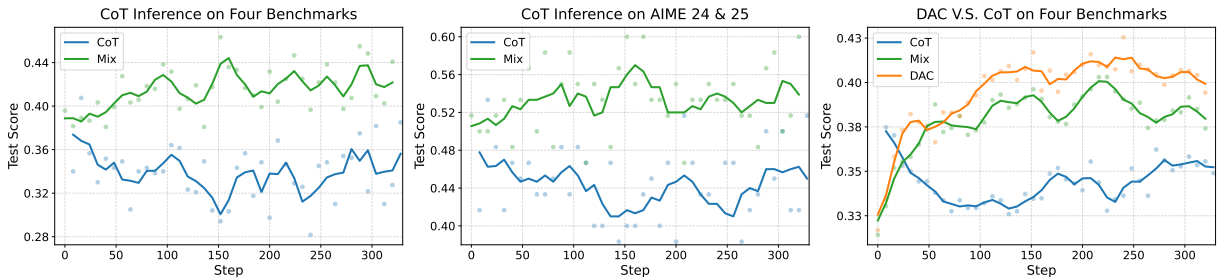


Figure 5: An overall comparison among standard CoT-RL, Mix-RL, and DAC-RL trained on all problems. **Left:** *Pass@1* accuracy on all benchmarks using CoT inference for both CoT-RL and Mix-RL. **Middle:** AIME score under the same settings as in *Left*. **Right:** Average performance across four competition-level benchmarks, where the CoT-trained policy uses CoT-style inference, while the other two employ DAC-style inference.

problem groups n and the number of conquering solutions per group m such that $n \times m = k$. As shown in Figure 6, allocating more groups (i.e., larger n and smaller m) consistently improves performance on competition-level benchmarks compared to the CoT baseline. This indicates that increased subproblem diversity at test time expands the model’s exploration space, improving its chances to discover correct trajectories.

4.3 Concise and Diverse Reasoning with DAC

Intuitively, guiding the model to generate responses to subproblems may introduce additional reasoning steps and increase output length. However, in our *Deep DAC* experiments, we observe that DAC training produces more concise reasoning than CoT-style RL. This results in fewer rollouts being clipped by the maximum length constraint, which may reduce false negatives arising from truncated positive rollouts, while lowers the

per-step training time of DAC-RL. An example with detailed analysis is present in Appendix F.

Despite producing more compact training rollouts, DAC exhibits broader exploration, as reflected by its higher policy entropy compared with CoT in the right panel of Figure 7. This indicates that DAC-style reasoning compaction does not compromise exploration; instead, it removes redundant steps while preserving—and even enhancing—diversity in solution space exploration, thereby enabling more efficient training without premature convergence (Cui et al., 2025).

5 Related Work

5.1 Divide-and-Conquer in LLM Reasoning

Divide-and-conquer is a fundamental algorithm design paradigm (Cormen et al., 2022). Its core idea is to decompose a complex problem into sub-problems, solve them first, and then combine their

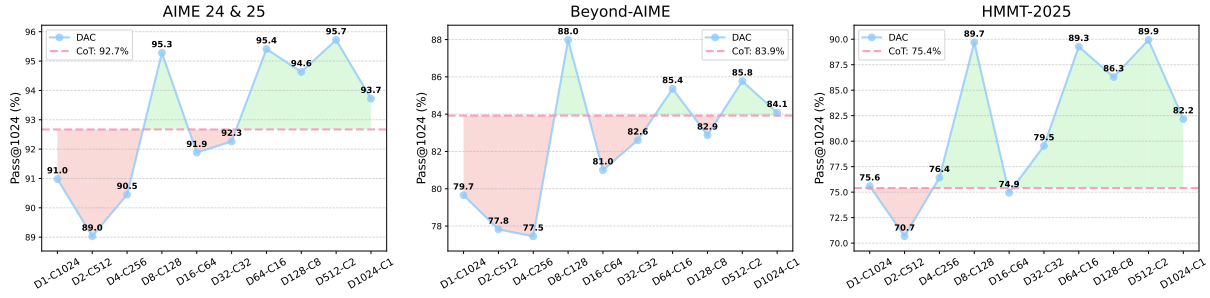


Figure 6: The $Pass@k$ performance under different allocations between *division* and *conquering* with a fixed budget total of $k = 1024$. The CoT baseline corresponds to 1024 independent generations.

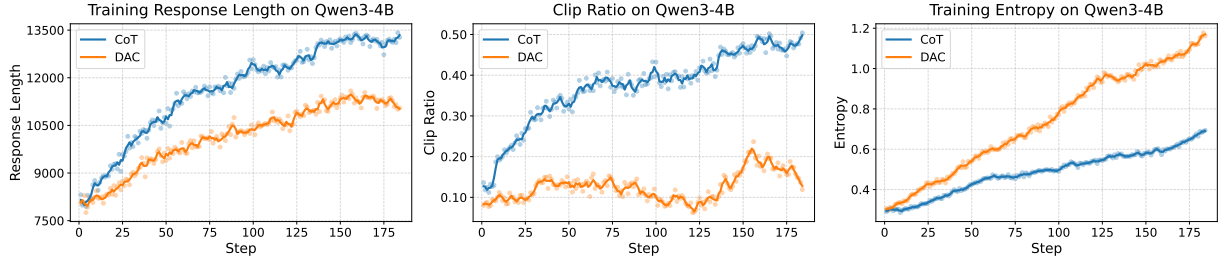


Figure 7: The batch-averaged response length, clip ratio of intermediate rollouts, and policy entropy in training.

solutions to obtain the final answer. In LLM reasoning, several studies (Zhou et al., 2022; Ling et al., 2023; Khot et al., 2022; Huang et al., 2022; Dua et al., 2022; Chen et al., 2022; Zelikman et al., 2023; Ye et al., 2023; Xue et al., 2024) focus on decomposing complex problems into subproblems using prompting-based strategies. For example, Least-to-Most Prompting (Zhou et al., 2022) incorporates a few in-context examples to guide LLMs in generating a sequence of simpler subproblems from the original one. Another line of work (Yao et al., 2023; Besta et al., 2024; Chen et al., 2022; Yang et al., 2024b) intrinsically guides LLMs to decompose reasoning beyond the standard CoT (Wei et al., 2022) through mechanisms such as expansion, search, or reflection.

Although effective, existing studies apply the DAC strategy only during inference, suffering from a misalignment between the DAC inference and general post-training, which primarily emphasizes direct problem answering (Yang et al., 2024a). Ladder (Simonds and Yoshiyama, 2025) proposes decomposing integral problems into simpler sub-tasks and incorporating the solutions into training. In this work, we train LLMs’ DAC reasoning with a unified RL framework and show that this paradigm offers a higher reasoning ceiling.

5.2 Reinforcement Learning for LLMs

Large-scale reinforcement learning (RL) has significantly improved LLMs in complex reasoning tasks (Luong et al., 2024; Guo et al., 2025). Al-

gorithms such as PPO (Schulman et al., 2017), GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025) have shown strong generalization and effectiveness in LLM post-training. Existing efforts in scaling up RL optimization have focused on enhancing exploration (Yu et al., 2025; Yuan et al., 2025; Liu et al., 2025; Yeo et al., 2025; Huang et al., 2025) and adapting RL to long CoT conditions (Jaech et al., 2024; Guo et al., 2025; Li et al., 2025; Yang et al., 2025). However, these methods overlook the limitation that models cannot improve on problems they consistently answer incorrectly under CoT reasoning, while we address this by integrating the DAC strategy into RL training, enabling the model to learn and solve them via subproblem decomposition.

6 Conclusion

In this paper, we conduct a comprehensive study of the divide-and-conquer (DAC) reasoning paradigm beyond standard CoT for LLMs in addressing complex reasoning problems. We identify a gap between models trained under general or CoT-centric post-training and their DAC reasoning capabilities at inference time, and propose enhancing DAC reasoning through RL-based training. The experiments shows that DAC-style reasoning demonstrates a higher performance upper bound and stronger test-time scalability than traditional step-by-step CoT, leading to improved performance on competition-level benchmarks.

7 Limitation

Despite the empirical effectiveness of our DAC reasoning framework, it relies on a fixed prompt set throughout training and evaluation, leaving room for further gains through improved prompt engineering. The experiments also require a sufficient level of instruction-following capability for the model to reliably decompose problems into meaningful subproblems. In addition, although we examine several reward designs based on conquering accuracy for guiding subproblem division, other more effective reward formulations may exist. Overall, adapting multiple prompt sets and investigating additional division reward settings could further enhance the robustness of our whole DAC reasoning framework.

References

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2025. Matharena: Evaluating llms on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690.

ByteDance-Seed. 2025. Beyondaime: Advancing math reasoning evaluation beyond high school olympiads. <https://huggingface.co/datasets/ByteDance-Seed/BeyondAIME>.

Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, and 1 others. 2025. Seed-prover: Deep and broad reasoning for automated theorem proving. *arXiv preprint arXiv:2507.23726*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2022. *Introduction to algorithms*. MIT press.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, and 1 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive prompting for decomposing complex questions. *arXiv preprint arXiv:2212.04092*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Fanding Huang, Guanbo Huang, Xiao Fan, Yi He, Xiao Liang, Xiao Chen, Qinting Jiang, Faisal Nadeem Khan, Jingyan Jiang, and Zhi Wang. 2025. Beyond the exploration-exploitation trade-off: A hidden state approach for llm reasoning in rlvr. *arXiv preprint arXiv:2509.23808*.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, and 1 others. 2025. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.

Xiao Liang, Zhongzhi Li, Yeyun Gong, Yelong Shen, Ying Nian Wu, Zhijiang Guo, and Weizhu Chen. 2025. Beyond pass@ 1: Self-play with variational problem synthesis sustains rlvr. *arXiv preprint arXiv:2508.14029*.

Yong Lin, Hangyu Lin, Wei Xiong, Shizhe Diao, Jianmeng Liu, Jipeng Zhang, Rui Pan, Haoxiang Wang, Wenbin Hu, Hanning Zhang, and 1 others. 2023. Mitigating the alignment tax of rlhf. *arXiv preprint arXiv:2309.06256*.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 36:36407–36433.

618	Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi,	An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao,	671
619	Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin.	Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong	672
620	2025. Understanding r1-zero-like training: A criti-	Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024a.	673
621	cal perspective. <i>arXiv preprint arXiv:2503.20783</i> .	Qwen2. 5-math technical report: Toward mathe-	674
622	Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng	matical expert model via self-improvement. <i>arXiv</i>	675
623	Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Rea-	<i>preprint arXiv:2409.12122</i> .	676
624	soning with reinforced fine-tuning. <i>arXiv preprint</i>	Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao,	677
625	<i>arXiv:2401.08967</i> , 3.	Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and	678
626	MAA. 2024. American invitational mathemat-	Bin Cui. 2024b. Buffer of thoughts: Thought-	679
627	ics examination (AIME). https://maa.org/	augmented reasoning with large language models.	680
628	math-competitions/aime .	<i>Advances in Neural Information Processing Sys-</i>	681
629	ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin,	<i>tems</i> , 37:113519–113544.	682
630	Haocheng Wang, Wanxia Zhao, Liyue Zhang, Zhe	Zhicheng Yang, Zhijiang Guo, Yinya Huang, Xiaodan	683
631	Fu, Qihao Zhu, Dejian Yang, and 1 others. 2025.	Liang, Yiwei Wang, and Jing Tang. 2025. Treerpo:	684
632	Deepseek-prover-v2: Advancing formal mathemati-	Tree relative policy optimization. <i>arXiv preprint</i>	685
633	cal reasoning via reinforcement learning for subgoal	<i>arXiv:2506.05183</i> .	686
634	decomposition. <i>arXiv preprint arXiv:2504.21801</i> .	Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran,	687
635	John Schulman, Filip Wolski, Prafulla Dhariwal,	Tom Griffiths, Yuan Cao, and Karthik Narasimhan.	688
636	Alec Radford, and Oleg Klimov. 2017. Proxi-	2023. Tree of thoughts: Deliberate problem solv-	689
637	mal policy optimization algorithms. <i>arXiv preprint</i>	ing with large language models. <i>Advances in neural</i>	690
638	<i>arXiv:1707.06347</i> .	<i>information processing systems</i> , 36:11809–11822.	691
639	Ning Shang, Yifei Liu, Yi Zhu, Li Lyna Zhang, Wei-	Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei	692
640	jiang Xu, Xinyu Guan, Buze Zhang, Bingcheng	Huang, and Yongbin Li. 2023. Large language mod-	693
641	Dong, Xudong Zhou, Bowen Zhang, and 1 others.	els are versatile decomposers: Decomposing evi-	694
642	2025. rstar2-agent: Agentic reasoning technical re-	dence and questions for table-based reasoning. In	695
643	port. <i>arXiv preprint arXiv:2508.20722</i> .	<i>Proceedings of the 46th international ACM SIGIR</i>	696
644	Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu,	<i>conference on research and development in informa-</i>	697
645	Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan	<i>tion retrieval</i> , pages 174–184.	698
646	Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseek-	Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neu-	699
647	math: Pushing the limits of mathematical reason-	big, and Xiang Yue. 2025. Demystifying long	700
648	ing in open language models. <i>arXiv preprint</i>	chain-of-thought reasoning in llms. <i>arXiv preprint</i>	701
649	<i>arXiv:2402.03300</i> .	<i>arXiv:2502.03373</i> .	702
650	Toby Simonds and Akira Yoshiyama. 2025. Ladder:	Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan,	703
651	Self-improving llms through recursive problem de-	Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu,	704
652	composition. <i>arXiv preprint arXiv:2503.00735</i> .	Lingjun Liu, Xin Liu, and 1 others. 2025. Dapo:	705
653	Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren,	An open-source llm reinforcement learning system	706
654	Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He,	at scale. <i>arXiv preprint arXiv:2503.14476</i> .	707
655	Kuan Wang, Jianfeng Gao, and 1 others. 2025. Re-	Yufeng Yuan, Qiyang Yu, Xiaochen Zuo, Ruofei Zhu,	708
656	inforcement learning for reasoning in large language	Wenyuan Xu, Jiase Chen, Chengyi Wang, TianTian	709
657	models with one training example. <i>arXiv preprint</i>	Fan, Zhengyin Du, Xiangpeng Wei, and 1 oth-	710
658	<i>arXiv:2504.20571</i> .	ers. 2025. Vapo: Efficient and reliable reinforc-	711
659	Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten	ment learning for advanced reasoning tasks. <i>arXiv</i>	712
660	Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou,	<i>preprint arXiv:2504.05118</i> .	713
661	and 1 others. 2022. Chain-of-thought prompting	Eric Zelikman, Qian Huang, Gabriel Poesia, Noah	714
662	elicits reasoning in large language models. <i>Ad-</i>	Goodman, and Nick Haber. 2023. Parsel: Algorith-	715
663	<i>advances in neural information processing systems</i> ,	mic reasoning with language models by composing	716
664	35:24824–24837.	decompositions. <i>Advances in Neural Information</i>	717
665	Shangzi Xue, Zhenya Huang, Jiayu Liu, Xin Lin, Yut-	<i>Processing Systems</i> , 36:31466–31523.	718
666	ing Ning, Binbin Jin, Xin Li, and Qi Liu. 2024.	Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei,	719
667	Decompose, analyze and rethink: Solving intricate	Nathan Scales, Xuezhi Wang, Dale Schuurmans,	720
668	problems with human-like reasoning cycle. <i>Ad-</i>	Claire Cui, Olivier Bousquet, Quoc Le, and 1 others.	721
669	<i>advances in Neural Information Processing Systems</i> ,	2022. Least-to-most prompting enables complex	722
670	37:357–385.	reasoning in large language models. <i>arXiv preprint</i>	723
		<i>arXiv:2205.10625</i> .	724

A Preliminaries on GRPO for LLMs

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is an efficient algorithm for reinforcement learning in LLMs, where the advantages for each token in a rollout are computed in a group-relative manner without requiring an additional critic model to estimate token values. Specifically, given an input prompt x , the policy model $\pi_{\theta_{\text{old}}}$ generates a group of G responses $\mathbf{Y} = \{y_i\}_{i=1}^G$, with acquired rewards $\mathbf{R} = \{r_i\}_{i=1}^G$. The advantage $A_{i,t}$ for each token in response y_i is computed as the group-normalized rewards:

$$A_{i,t} = \frac{r_i - \text{mean}(\{r_i\}_{i=1}^G)}{\text{std}(\{r_i\}_{i=1}^G)}. \quad (5)$$

To improve the stability of policy optimization, GRPO clips the probability ratio $k_{i,t}(\theta) = \frac{\pi_{\theta}(y_{i,t}|x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x, y_{i,<t})}$ within a trust region (Schulman et al., 2017), and constrains the policy distribution from deviating too much from the reference model using a KL term. The final optimization objective is defined as follows:

$$\mathcal{J}(\theta) = \mathbb{E}_{x \sim \mathcal{D}, \mathbf{Y} \sim \pi_{\theta_{\text{old}}}(\cdot|x)} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \left(\min \left(k_{i,t}(\theta) A_{i,t}, \text{clip}(k_{i,t}(\theta), 1 - \varepsilon, 1 + \varepsilon) A_{i,t} \right) - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}) \right) \right] \quad (6)$$

In this work, we incorporate several techniques from (Yu et al., 2025) into training, including Clip-Higher and Token-Level Loss, all of which are widely adopted to enhance training efficiency.

B Proof for Lemma 2.1

Let $s_i \in \{0, 1\}$ indicate whether the solution to subproblem i is correct, and let $\mathbf{s} = (s_1, \dots, s_m)$ denote a realization of the random vector $S = (S_1, \dots, S_m)$. Let $C \in \{0, 1\}$ indicate whether the final answer to the original problem is correct. The *conquering* inference first sequentially solves the m subproblems and then addresses the original problem, thereby inducing the causal structure $\mathbf{s} \rightarrow C$, where subproblem correctness determines the correctness of the final solution.

Assumption 1: Monotonicity. $P(C = 1 | S = \mathbf{s})$ is nondecreasing in each coordinate of \mathbf{s} and strictly increasing in at least one coordinate. Intuitively, holding all other subproblem outcomes fixed, solving one additional subproblem correctly cannot decrease the probability of producing a correct final answer.

Assumption 2: Policy control. The policy π_{θ} determines the distribution of subproblem correctness, i.e., $P_{\theta}(S = \mathbf{s})$ depends on θ .

Proof. To analyze how optimizing for $C = 1$ affects subproblem correctness, we expand $P_{\theta}(C = 1)$ using the law of total probability:

$$P_{\theta}(C = 1) = \sum_{\mathbf{s} \in \{0,1\}^m} P(C = 1 | S = \mathbf{s}) P_{\theta}(S = \mathbf{s}) = \mathbb{E}_{S \sim P_{\theta}}[g(S)], \quad g(\mathbf{s}) := P(C = 1 | S = \mathbf{s}).$$

Under **Assumption 1**, g is an increasing function. Therefore, any policy update that increases $P_{\theta}(C = 1)$ must, on average, allocate more probability mass to configurations with higher numbers of correct subproblems. For any i , Bayes' rule gives:

$$P_{\theta}(S_i = 1 | C = 1) = \frac{\sum_{\mathbf{s}: s_i=1} P(C = 1 | S = \mathbf{s}) P_{\theta}(S = \mathbf{s})}{P_{\theta}(C = 1)}.$$

Because $P(C = 1 | S = \mathbf{s})$ is larger whenever $s_i = 1$ (by **Assumption 1**), it follows that

$$P_{\theta}(S_i = 1 | C = 1) \geq P_{\theta}(S_i = 1),$$

with strict inequality whenever S_i influences C . Equivalently,

$$\text{Cov}_{\theta}(\mathbf{1}\{S_i = 1\}, \mathbf{1}\{C = 1\}) \geq 0.$$

762 Hence, conditioning on (and rewarding) $C = 1$ preferentially upweights trajectories with more correct
763 subproblems. Under **Assumption 1-2**, the scalar reward

$$764 \quad \mathbf{R}(y_c) = \mathbf{1}\{\text{Extract}(y_c) = a\}, \quad \mathbb{E}_{y_c \sim \pi_\theta}[\mathbf{R}(y_c)] = P_\theta(C = 1),$$

765 serves as a consistent surrogate for subproblem correctness. The converse also holds for $C = 0$, where
766 incorrect final answers correspond to lower subproblem correctness.

C Cold-Start Distillation for DAC-RL

One concern when applying smaller models to DAC-RL training, as in our main experiments in Section 3.2, is their limited instruction-following capacity, which may hinder exploration during solution generation and policy optimization. Therefore, in this section, we investigate whether cold-starting DAC-style reasoning via distillation from a stronger instruction model can benefit subsequent RL training, and compare its performance with a CoT baseline under comparable settings. For distillation, we randomly select 3K problems from the DeepMath-103K dataset with difficulty labels greater than six and use a stronger instruction-following model, Qwen3-235B-A22B-2507-Instruct, to generate both CoT and DAC-style solutions, yielding a total of 6K problem–response pairs. We directly mix the CoT and DAC distillation data to form the training set, use Qwen3-4B-Instruct-2507 for the experiment, set the maximum token length to 32K, and fine-tune the model for five epochs.

The experimental results are shown in Table 2, where we report model performance both after cold-starting and after subsequent RL fine-tuning. Overall, cold-start improves the performance of both DAC and CoT. Under an equal budget of 3k samples, the distilled model exhibits stronger DAC-style reasoning performance than CoT, even though the initial model performs better with CoT. After RL fine-tuning, the DAC model achieves a larger performance gain than the CoT baseline, with improvements increasing from 1.4% to 2.4%. The consistent improvements across both training paradigms indicate that DAC constitutes a more effective advanced reasoning paradigm. Moreover, the amplified gains observed after RL suggest that DAC enables richer and more diverse exploration during policy optimization, leading to more efficient performance improvement than standard CoT reasoning.

D The Effects of a Subproblem-Solving Format Constraint

In this section, we explore the impact of imposing a constraint that enforces the *conquering* solutions to explicitly solve subproblems one by one, assigning positive rewards only to solutions that adhere to this rule. Specifically, we require the responses to include all “subproblem i ” for $i = 1, \dots, n_g - 1$, where n_g denotes the number of

Generate Solution Steps Instead of Subproblems

>>> Problem:

Add 53.463 to 12.9873 and round to the nearest thousandth.

>>> Divide Response:

<SUBPROBLEM 1>

This subproblem is about adding the two numbers: 53.463 and 12.9873.

$53.463 + 12.9873 = 66.4503$

</SUBPROBLEM 1>

<SUBPROBLEM 2>

Now, we need to round 66.4503 to the nearest thousandth. The thousandth place is the third digit after the decimal point. Looking at the fourth digit (which is 3 in this case), we determine if we need to round up or stay the same.

66.4503 rounded to the nearest thousandth is 66.450 since the fourth digit (3) is less than 5.

</SUBPROBLEM 2>

<SUBPROBLEM 3>

So, rounding 66.4503 to the nearest thousandth gives us 66.450.

</SUBPROBLEM 3>

<SUBPROBLEM 4>

We have confirmed that 66.4503 rounded to the nearest thousandth is 66.450.

The final answer is 66.450.

</SUBPROBLEM 4>

<SUBPROBLEM 5>

Thus, the final rounded number is 66.450.

</SUBPROBLEM 5>

Figure 8: A case that the policy generates intermediate solution steps instead of dividing subproblems.

subproblems generated during the divide stage. A response receives a positive reward only if it adheres to the constrained format with a correct final answer; otherwise, it receives a negative reward.

The results are reported in Table 2. Although the policy learns to explicitly answer all subproblems in a strict format—unlike models without this constraint—its evaluation performance is worse than models without this constraint. This finding aligns with the conclusions in (Lin et al., 2023; Wang et al., 2025) that training LLMs for alignment, especially under strict formatting constraints, can introduce an *alignment tax*, which enhances instruction-following behavior while degrading downstream performance.

E Discussion on Conquering Accuracy as Feedback for Division Reward

Intuitively, rewarding the divided subproblems based on how effectively they help the policy solve the original problem encourages the model

Model	AIME 2024		AIME 2025		Beyond-AIME		HMMT 2025		Average	
	Pass@1	Pass@32	Pass@1	Pass@32	Pass@1	Pass@32	Pass@1	Pass@32	Pass@1	Pass@32
Init-CoT	62.6	90.0	45.7	76.7	32.1	65.0	30.3	56.7	42.7	72.1
Init-DAC	59.6	90.0	43.2	73.3	29.6	61.0	28.2	63.3	40.2	71.9
CD-CoT	58.0	90.0	49.0	80.0	34.9	66.0	31.9	63.3	43.4	74.8
CD-DAC	60.2	86.7	53.5	83.3	35.9	68.0	35.9	66.7	46.4	76.2
CD-RL-CoT	66.5	92.5	60.5	88.2	41.2	66.1	40.5	71.8	52.2	79.7
CD-RL-DAC	71.3	91.7	61.9	91.7	41.5	69.0	39.3	75.9	53.5	82.1
Δ (RL)	+4.8	-0.8	+1.4	+3.5	+0.3	+2.9	-1.2	+4.1	+1.3	+2.4

Table 2: Experiments with cold-start (CD) initialization. For rows labeled “CD,” both CoT and DAC models are initialized via supervised distillation using the same amount of training samples, and models marked with “R” further undergo reinforcement learning as described in Section 2.

Model	Format Following	AIME 2024		AIME 2025		Beyond-AIME		HMMT 2025		Average	
		Pass@1	Pass@32	Pass@1	Pass@32	Pass@1	Pass@32	Pass@1	Pass@32	Pass@1	Pass@32
Init	72.1%	60.9	86.7	44.6	76.7	30.4	62.0	27.0	63.3	40.7	72.2
\hookrightarrow RL w.o. Format	42.6%	66.3	91.6	61.5	87.6	38.8	70.7	38.7	76.4	51.3	81.6
\hookrightarrow RL w.i. Format	92.1%	56.2	86.3	55.5	87.1	34.3	66.6	34.6	69.2	45.2	77.3

Table 3: Evaluating the alignment tax from enforcing strict subproblem answer formats in conquering solutions on the format following rate and model performance. *w.i. Format* refers to the imposition of the constraint during RL.

to generate higher-quality subproblems. Motivated by this intuition, our initial implementation adopts a straightforward strategy: using the average accuracy of the G_c conquer solutions given the subproblem group \mathcal{P}_g as the reward for divide responses instead of Equation 2, denoted as $\frac{1}{|G_c|} \sum_{\{y_{g,v}\}_{v=1}^{G_c}} \{\mathbf{R}(y_{g,v}) \mid \mathcal{P}_g\}_{v=1}^{G_c}$. However, the experiments show that when conquer-accuracy-based rewards are used to reward subproblems division, the model tends to solve the problem prematurely during the *division* stage (as illustrated in Case 8), rather than decomposing it into subproblems. This indicates that when the model discovers outputs that better support subsequent conquer-stage solution generation—i.e., yield higher rewards—it may ignore the instructions specified in the prompt. *More generally, these results suggest that when prompt instructions substantially conflict with the reward assignment scheme in RL training, the policy tends to prioritize reward maximization over strict adherence to the instructions, highlighting the importance of aligning prompt design with reward specification in RL settings.* This initial failure motivates us to adopt a relaxed division reward that guarantees only a lower bound on subproblem helpfulness, rather than strictly optimizing for exact accuracy, as formalized in Equation 2. This relaxation reduces greedy behavior in early training and prevents the policy from prematurely optimizing for the original problem, thereby preventing this failure mode.

F A Case Study on How DAC Reduces Reasoning Redundancy

In this section, we present a case study based on a problem from the DAPO-Math-17k dataset. The original problem and its decomposed subproblems under DAC reasoning are shown in Figure 9. As the full responses of both reasoning styles are lengthy, we present partial solutions targeting the same linear system to clearly distinguish the two reasoning styles and highlight the compression effect of DAC. Partial reasoning traces produced by DAC and CoT are shown in Figures 10 and 11, respectively, with the corresponding full solutions in Listings 1 (3,328 tokens) and 2 (5,072 tokens).

Both DAC and CoT address the same algebraic structure by introducing auxiliary variables to linearize and solve the original system. However, the DAC sub-solution adheres to a predefined subproblem decomposition, mapping each subproblem directly to a necessary transformation or computation, thereby avoiding redundant restatements and self-corrections. In contrast, CoT reasoning involves *repeated verification and re-derivation* of intermediate results, which leads to longer and less structured outputs. Consequently, DAC compresses reasoning primarily by reducing narrative redundancy rather than simplifying the underlying mathematics, resulting in shorter, more stable, and more token-efficient reasoning traces.

The Original Problem and Divided Subproblems

>>> Original Problem:

Let a, b, c be positive real numbers for which:

$$\frac{5}{a} = b + c, \quad \frac{10}{b} = c + a, \quad \text{and} \quad \frac{13}{c} = a + b.$$

If $a + b + c = \frac{m}{n}$ for relatively prime positive integers m and n , compute $m + n$.

>>> Divided Subproblems:

<SUBPROBLEM 1>

Rewrite the given equations into an equivalent algebraic system involving a, b , and c .

</SUBPROBLEM 1>

<SUBPROBLEM 2>

Transform the system into a simpler form by introducing appropriate substitutions or auxiliary variables.

</SUBPROBLEM 2>

<SUBPROBLEM 3>

Solve the resulting system to determine the values of the substituted variables.

</SUBPROBLEM 3>

<SUBPROBLEM 4>

Recover the values of a, b , and c from the solved variables.

</SUBPROBLEM 4>

<SUBPROBLEM 5>

Compute $a + b + c$ from the obtained solution and express it in lowest terms.

</SUBPROBLEM 5>

Now use individual equations:

$$-x + z = 5 \rightarrow x = 5 - z$$

$$- \text{From this, plug into } x + y = 10: 5 - z + y = 10 \rightarrow y = 5 + z$$

$$- \text{Now use } y + z = 13: \text{ left side becomes } 5 + z + z = 5 + 2z = 13$$

$$\rightarrow 2z = 8 \rightarrow z = 4$$

$$\text{Then } x = 5 - z = 1, y = 5 + 4 = 9$$

So:

$$- ab = x = 1$$

$$- bc = y = 9$$

$$- ac = z = 4$$

Figure 10: A partial solution produced during DAC conquering for the problem illustrated in Figure 9.

Partial Response from CoT Reasoning

So now, we have:

$$\text{From first: } ab + ac = 5$$

$$\text{From second: } ab + bc = 10$$

$$\text{From third: } bc + ca = 14? \text{ Wait no.}$$

Wait solve carefully:

$$\text{Wait first: } ab + ac = 5$$

$$\text{Second: } ab + bc = 10$$

$$\text{Third: from } c(a + b) = 13: c(a + b) = ca + cb = 13$$

So indeed:

$$- ab + ac = 5 \rightarrow \text{(i)}$$

$$- ab + bc = 10 \rightarrow \text{(ii)}$$

$$- bc + ca = 13 \rightarrow \text{(iii)}$$

$$\text{But wait — we previously found } ab + bc + ca = 14$$

Wait now we have:

Add all three:

$$\text{(i)+(ii)+(iii):}$$

$$\text{Left: } ab + ac + ab + bc + bc + ca$$

$$= 2ab + 2bc + 2ca$$

So:

$$2(ab + bc + ca) = 5 + 10 + 13 = 28$$

$$\text{So } 2(ab + bc + ca) = 28 \rightarrow ab + bc + ca = 14 \rightarrow \text{matches what we got earlier.}$$

Good — consistent.

Now, let's label:

Let:

$$- x = ab$$

$$- y = bc$$

$$- z = ca$$

Then from above:

$$- x + z = 5 \rightarrow \text{(from } ab+ac=ab+ac=x+z=5)$$

$$- x + y = 10$$

$$- y + z = 13$$

$$\text{And } x + y + z = 14$$

Now solve this system.

Figure 9: The original problem from DAPO-Math-17k and the decomposed subproblems in DAC reasoning.

Partial Response from DAC Reasoning

Now denote the variables: let:

$$- x = ab$$

$$- y = bc$$

$$- z = ac$$

But notice that now:

$$- (1') \rightarrow ab + ac = x + z = 5$$

$$- (2') \rightarrow ab + bc = x + y = 10$$

$$- (3') \rightarrow ac + bc = z + y = 13$$

Now we have a simpler linear system in terms of x, y, z :

$$\begin{cases} x + z = 5 \\ x + y = 10 \\ y + z = 13 \end{cases}$$

Add all three equations:

$$\text{Left-hand side: } x + z + x + y + y + z = 2x + 2y + 2z = 2(x + y + z)$$

$$\text{Sum of RHS} = 5 + 10 + 13 = 28$$

Thus:

$$2(x + y + z) = 28 \Rightarrow x + y + z = 14$$

From:

- (1) $x + z = 5$
- (2) $x + y = 10$
- (3) $y + z = 13$

Subtract (1) from (2): $(2)-(1): (x + y) - (x + z) = 10 - 5 \Rightarrow y - z = 5$

From (3): $y + z = 13$

Now solve:

From (3) and $(y-z=5)$:

Add them: $(y + z) + (y - z) = 13 + 5 = 18 \Rightarrow 2y = 18 \Rightarrow y = 9$

Then from $y + z = 13$, $z = 13 - 9 = 4$

From (2): $x = 10 - y = 10 - 9 = 1$

From (1): $x + z = 1 + 4 = 5$: yes, checks.

So:

- $ab = 1$
- $bc = 9$
- $ca = 4$

Nice. So now we know:

- $ab = 1$
- $bc = 9$
- $ca = 4$

Figure 11: A partial solution generated by CoT-style reasoning for the problem illustrated in Figure 9.

902
903
904
905
906
907
908
909
910

G Prompts

We present the prompts used in our framework, including the *division* prompt and the *conquering* prompt, in Figures 12 and 13. During both RL training and inference, the subproblem placeholders in the *conquering* prompt are replaced with the actual decomposed subproblems, following the format “<SUBPROBLEM i> . . . </SUBPROBLEM i>” as illustrated in Figure 13.

The Division Prompt

You are given the following mathematical problem:

{REPLACE}

Your task is NOT to solve the problem. Instead, break it down into **more than 3 subproblems** that, if solved, would naturally lead to the solution of the original problem.

Please follow this output format strictly (include more than 3 subproblems as appropriate):

```
<SUBPROBLEM 1>
<first subproblem>
</SUBPROBLEM 1>

<SUBPROBLEM 2>
<second subproblem>
</SUBPROBLEM 2>

<SUBPROBLEM 3>
<third subproblem>
</SUBPROBLEM 3>

<SUBPROBLEM 4>
<fourth subproblem, if needed>
</SUBPROBLEM 4>

<SUBPROBLEM 5>
<fifth subproblem, if needed>
</SUBPROBLEM 5>
```

Do not provide any explanations or answers. Only list the subproblems using the tags <SUBPROBLEM x> and </SUBPROBLEM x> as shown.

Figure 12: The *division* prompt for dividing the original problem into subproblems.

The Conquering Prompt

You are given a challenging math problem and its decomposition into subproblems.

Your task is to:

1. Solve the **subproblems sequentially** (in the given order).
2. Show detailed reasoning for each subproblem.
3. **Do not use ‘\boxed’** when solving subproblems.
4. After solving all subproblems, combine their results to solve the **original problem**.
5. Present the final answer to the original problem **only once**, enclosed in ‘\boxed{ }’.

Input

Original Problem:
{REPLACE}

Subproblems:

```
<SUBPROBLEM 1>
<first subproblem>
</SUBPROBLEM 1>

<SUBPROBLEM 2>
<second subproblem>
</SUBPROBLEM 2>

<SUBPROBLEM 3>
<third subproblem>
</SUBPROBLEM 3>

<SUBPROBLEM 4>
<fourth subproblem, if needed>
</SUBPROBLEM 4>

<SUBPROBLEM 5>
<fifth subproblem, if needed>
</SUBPROBLEM 5>
—
```

Final Instruction

- * First, go through the subproblems one by one with detailed reasoning.
- * Finally, derive the answer to the **original problem** and present it in the format:

```
$$
\boxed{\text{final answer here}}
$$
```

Figure 13: The *conquering* prompt for solving the subproblems and the original problem.

911

912

Listing 1: The full solution generated during DAC conquering for the problem illustrated in Figure 9.

```

913
914 We are given that $ a, b, c > 0 $ are real numbers
915 satisfying:
916
917 $$
918 \frac{5}{a} = b+c \tag{1}
919 $$
920 $$
921 \frac{10}{b} = c+a \tag{2}
922 $$
923 $$
924 \frac{13}{c} = a+b \tag{3}
925 $$
926
927 Our goal is to determine $ a + b + c $ in the form $
928 \frac{m}{n} $ with $ \gcd(m, n) = 1 $, then compute
929 $ m+n $.
930
931 ---
932
933 ## <SUBPROBLEM 1> Rewrite the given equations into
934 an equivalent algebraic system involving $a$, $b$,
935 and $c$.
936
937 Start by treating this system algebraically as a
938 consistent system with three equations and three
939 variables.
940
941 We are to solve for three positive real numbers $ a,
942 b, c $ satisfying:
943
944 $$
945 (b+c)a = 5 \Rightarrow ab + ac = 5 \tag{1'}
946 $$
947 $$
948 (c+a)b = 10 \Rightarrow bc + ab = 10 \tag{2'}
949 $$
950 $$
951 (a+b)c = 13 \Rightarrow ac + bc = 13 \tag{3'}
952 $$
953
954 Now we have the symmetric-looking expressions.
955
956 We have:
957
958 (1') $ ab + ac = 5 $
959
960 (2') $ ab + bc = 10 $
961
962 (3') $ ac + bc = 13 $
963
964 Now denote the variables: let:
965
966 - $ x = ab $
967 - $ y = bc $
968 - $ z = ac $
969
970 But notice that now:
971 - (1') -> $ ab + ac = x + z = 5 $
972 - (2') -> $ ab + bc = x + y = 10 $
973 - (3') -> $ ac + bc = z + y = 13 $
974
975 Now we have a simpler linear system in terms of $ x,
976 y, z $:
977
978 $$
979 \begin{cases}
980 x + z = 5 \\
981 x + y = 10 \\
982 y + z = 13 \end{cases}
983 \end{cases}
984 $$
985
986 Add all three equations:
987
988 Left-hand side: $ x + z + x + y + y + z = 2x + 2y +
989 2z = 2(x+y+z) $
990
991 Sum of RHS = $ 5 + 10 + 13 = 28 $
992
993 Thus:
994 $$
995 2(x+y+z)=28 \Rightarrow x+y+z=14
996 $$
997
998 Now use individual equations:
999
1000 - $ x + z = 5 $ -> $ x = 5 - z $
1001 - From this, plug into $ x + y = 10 $: $ 5 - z + y =
1002 10 $ -> $ y = 5 + z $
1003 - Now use $ y + z = 13 $: left side becomes $ 5 + z
1004 + z = 5 + 2z = 13 $
1005 -> $ 2z = 8 $ -> $ z = 4 $
1006

```

Then \$ x = 5 - z = 1 \$, \$ y = 5 + 4 = 9 \$

So:

```

- $ ab = x = 1 $
- $ bc = y = 9 $
- $ ac = z = 4 $

```

Now use these derived values.

From:

```

- $ ab = 1 $ -> $ b = \frac{1}{a} $
- $ ac = 4 $ -> $ c = \frac{4}{a} $
- $ bc = 9 $

```

Let's now compute \$ bc \$ using those:

\$ b = 1/a \$, \$ c = 4/a \Rightarrow bc = (1/a)(4/a) = 4 / a^2 \$

But from equation, \$ bc = 9 \Rightarrow \frac{4}{a^2} = 9 \Rightarrow a^2 = \frac{4}{9} \Rightarrow a = \frac{2}{3} \$, since \$ a > 0 \$

Now we can get:

```

- $ b = \frac{1}{a} = \frac{1}{2/3} = \frac{3}{2} $
- $ c = \frac{4}{a} = \frac{4}{2/3} = 4 \cdot \frac{3}{2} = 6 $

```

Check if all equations hold

Now compute:

1. \$ \frac{5}{a} = 5 / (2/3) = 5 \cdot 3/2 = 15/2 = 7.5 \$

Left: \$ b + c = 3/2 + 6 = 3/2 + 12/2 = 15/2 \$ -> OK

2. \$ \frac{10}{b} = 10 \div (3/2) = 10 \cdot 2 / 3 = 20/3 \approx 6.666 \$

Left: \$ c + a = 6 + 2/3 = (18+2)/3 = 20/3 \$ -> OK

3. \$ \frac{13}{c} = 13 / 6 \approx 2.1666 \$

Left: \$ a + b = 2/3 + 3/2 = \$ find common denominator:

LCM of 3 and 2 is 6 -> \$ \frac{4}{6} + \frac{9}{6} = 13/6 \$ -> Yes

So left = 13/6, right = 13 / 6 -> OK

Perfect.

Final values:

```

$$
a = \frac{2}{3}, \quad b = \frac{3}{2}, \quad c = 6
$$

```

Now compute \$ a + b + c \$:

Compute sum:

First write everything over 6:

```

- $ a = 2/3 = 4/6 $
- $ b = 3/2 = 9/6 $
- $ c = 6 = 36/6 $

```

Sum: \$ \frac{4+9+36}{6} = \frac{49}{6} \$

So \$ a + b + c = \frac{49}{6} \$

Now since \$ \gcd(49, 6) = 1 \$, this fraction is already reduced.

Then \$ m = 49, n = 6 \$ -> so answer is \$ m+n = 49 + 6 = 55 \$

Wait - this seems to directly solve the problem.

But to ensure completeness, let's go through the other subproblems to confirm structure and uniqueness.

<SUBPROBLEM 2> Transform the system into a simpler form by introducing appropriate substitutions or auxiliary variables.

This question is really looking to see whether an interpretation in geometric terms, such as triangle-like inequalities, symmetry structures, monotonicity, etc., exists.

1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103

1104 But observe:
1105
1106 The equations involve symmetry broken by
1107 coefficients: 5,10,13 -> not symmetric.
1108
1109 Equations like:
1110
1111 $- \frac{5}{a} = b + c \rightarrow b + c > 0$ (always true
1112), and $(b + c)$ is inversely proportional to $a \rightarrow$ so
1113 if a is small, $b + c$ is large \rightarrow but constrained.
1114
1115 Interestingly, we have:
1116
1117 Each variable appears in an inversion (via
1118 reciprocal), such as $\frac{1}{a}$ is proportional to $(b +$
1119 $c)$. So all variables appear in such reciprocal-sum
1120 form.
1121
1122 We might interpret this as a symmetric-looking "
1123 harmonic-type" system.
1124
1125 Think about coefficients:
1126
1127 If a, b, c were symmetrically related, total
1128 expressions might balance.
1129
1130 But here the constants: 5,10,13 \rightarrow increasing \rightarrow
1131 related to $\frac{5}{a}, \frac{10}{b}, \frac{13}{c}$
1132 $\frac{5}{a}$
1133
1134 So the reciprocals have degrees increasing with
1135 index.
1136
1137 Alternatively, for such problems involving harmonic
1138 variables, it's common to transform into symmetric
1139 forms.
1140
1141 In fact, our solution found in subproblem 1
1142 completely worked out algebraically.
1143
1144 So geometric or structural insight might not be
1145 essential; instead this problem is algebraic.
1146
1147 But note the **structure**:
1148
1149 Define $ab + ac = 5$, etc. - each pair of products
1150 involving product between two variables equals
1151 something.
1152
1153 Indeed, such expressions (sums of pairwise products)
1154 with three variables suggest symmetric systems.
1155
1156 That's why we used substitutions.
1157
1158 In such systems, under positivity, solutions may be
1159 unique.
1160
1161 Moreover, as the functions involved are rational,
1162 and the equations are algebraic, and only one
1163 solution exists (we found one via direct
1164 substitution), we may expect uniqueness.
1165
1166 But we'll confirm.
1167
1168 This soul-bond between pairs may suggest a symmetric
1169 substitution.
1170
1171 Note that when we treated in terms of ab, bc, ca , we
1172 got linear equations - very structured.
1173
1174 So the structure is algebraic and symmetric in form
1175 with triple product structure. In particular, the
1176 system is symmetric under reindexing where the input
1177 constraints (5,10,13) are associated with a, b, c
1178 respectively - not symmetric.
1179
1180 So not symmetric.
1181
1182 But the symmetric-like pairing ($ab + ac, ab + bc,$
1183 $ac + bc$) does suggest a linear algebraic structure.
1184
1185 We derived the three values uniquely under $a > b >$
1186 c (verify: $a = 2/3 = 0.67, b = 1.5, c = 6 \rightarrow c > b >$
1187 $a \rightarrow$ actually monotonically decreasing in value),
1188 and all values positive, so uniquely determined via
1189 constraints.
1190
1191 So, structural constraints include:
1192
1193 - All variables $> 0 \rightarrow$ positivity applies strictly
1194 because all terms positive.
1195
1196 - Substitution leads to a system solvable via linear
1197 system in products.
1198
1199 - Each of the pairwise products has unique
1200 definition under positivity \rightarrow forcing unique

solution.
So no ambiguity.
Note also that the total sum $S = a + b + c$ is
uniquely determined.
So there can be only one such positive solution.
So geometric picture - maybe not meaningful?
Alternatively, perhaps there's a physical
interpretation: like resistors or harmonic mean -
but seems forced.
Rather than deep geometry, we see the equations
impose independent constraints.
So key structural insight:
- The problem is constrained to three variables with
three equations involving symmetric pairwise
products via linear conversion.
- The constraints mix variables in such ways that
the variables can be uniquely solved under
positivity.
Also, each expression is proportional to sum of the
other two variables:
- E.g., $\frac{5}{a} = b + c \rightarrow$ so b and c are parts forming
 $\frac{5}{a} \rightarrow$ similar to "sum of two variables is
reciprocal"
This suggests that each sum is related to the
reciprocal of variable.
Think of it in terms of harmonic means:
We know $\frac{1}{x} = \frac{1}{a}$, and $b + c \propto \frac{5}{a} \rightarrow \frac{1}{x} = \frac{1}{a} \propto b + c$
This suggests some harmonic-like character (a
variable's reciprocal is linear in sum of others),
hence it's similar to dual-type identities.
Nothing strong geometric, so likely algebraic in
nature and solved directly as done.

<SUBPROBLEM 3> Solve the resulting system to
determine the values of the substituted variables.
Note that we are only asked for such an a, b, c :
and it is uniquely determined under positivity,
since:
Since all variables are positive and the constraint
equations are algebraic and fully determined, **the**
solution must be unique.
Moreover, the function $f(a, b, c) = a + b + c$ is
strictly minimized or maximized not really, because
we are asked to find one such triple, and it's
uniquely determined.
Moreover, if we tried to perturb any variable
slightly, the constraint will be broken - because
the functions are strictly decreasing or increasing
nonlinearly.
For example: The function $\frac{5}{a}$ is
strictly decreasing in a , while sum $b + c$ is
increasing when a falls.
But in practice, we have determined a stable point.
So this triple is structural - the only satisfying
one for positive reals.
Therefore, minimal or maximal configuration - such
as minimal sum? But already the sum is uniquely
fixed.
So parts:
- There is **only one** feasible solution for positive
reals, hence no minimal or maximal; only one.
So such configurations are unique.
Hence, under constraints, only one such triple.
So answer so far is well-defined and unique.

1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297

```

1298 ---
1299
1300 ## <SUBPROBLEM 4> Recover the values of $a$, $b$,
1301 and $c$ from the solved variables.
1302
1303 Done completely above, via algebra.
1304
1305 We found:
1306 - $ a = \frac{2}{3}, b = \frac{3}{2}, c = 6 $
1307
1308 We already verified the constraints and the sum is:
1309
1310 $$
1311 a + b + c = \frac{2}{3} + \frac{3}{2} + 6
1312 $$
1313
1314 We can compute step-by-step:
1315
1316 Convert all to sixths:
1317
1318 - $ \frac{2}{3} = \frac{4}{6} $
1319 - $ \frac{3}{2} = \frac{9}{6} $
1320 - $ 6 = \frac{36}{6} $
1321
1322 Add -> $ (4+9+36)/6 = 49 /6 $
1323
1324 Thus:
1325
1326 $ a + b + c = \frac{49}{6} $
1327
1328 So the sum is $ \frac{49}{6} $, a fraction in lowest
1329 terms.
1330
1331 So this is the requested value for $ m, n $.
1332
1333 ---
1334
1335 ## <SUBPROBLEM 5> Compute $a+b+c$ from the obtained
1336 solution and express it in lowest terms.
1337
1338 We address uniqueness.
1339
1340 We have 3 equations in 3 variables.
1341
1342 The equations were:
1343
1344 (1): $ ab + ac = 5 $
1345
1346 (2): $ ab + bc = 10 $
1347
1348 (3): $ ac + bc = 13 $
1349
1350 In effect: three equations involving the symmetric
1351 products $ ab, bc, ac $.
1352
1353 We solved the linear system for $ ab, bc, ac $ -> that
1354 system was triangular, with unique solution.
1355
1356 Then from $ ab=1, bc=9, ac=4 $, and from symmetry of
1357 variables, we derived only one positive solution (
1358 all variables positive), due to unilateral
1359 constraints (only positive reals allowed).
1360
1361 Suppose we try a negative variable -> but variables
1362 are defined as positive -> all constraints violated.
1363
1364 For instance, if one of the variables were negative,
1365 $ b+c $ could be negative or positive - but $ \frac{5}{a} > 0 $, so RHS must be positive -> so each side
1366 is positive -> so $ b+c > 0 $ etc.
1367
1368 So all values are forced to be positive.
1369
1370 And each equation contains two variables in products
1371 (whose sum is linear), so movement in one variable
1372 entails constraints on others.
1373
1374 Now, we derived all entries from linear system
1375 uniquely.
1376
1377 So the entire system, given positivity and real
1378 numbers, has only one such triple.
1379
1380 Therefore:
1381
1382 - **Uniqueness**: yes - due to linearly independent
1383 equations in the pairwise products; and the
1384 positivity constraints.
1385
1386 - **Existence**: we have one such triple.
1387
1388 - Conditions: positivity, and algebraic equation
1389 solvability as above.
1390
1391 Hence, the solution is uniquely determined.
1392
1393 ---
1394

```

```

## Final Computation

We have found that, under the constraints, the only
triple of positive reals is:

$ a = \frac{2}{3}, b = \frac{3}{2}, c = 6 $

Sum: $ \frac{49}{6} $

Express as reduced fraction: already reduced since $
\gcd(49,6)=1 $

So $ m = 49, n = 6 $, $ m+n = 55 $

---

$$
\boxed{55}
$$

```

Listing 2: The full solution generated by CoT-style reasoning for the problem illustrated in Figure 9.

```

We are given the following system of equations with
positive real numbers $ a, b, c $:

$$
\frac{5}{a} = b + c \quad \textit{(1)}
$$
$$
\frac{10}{b} = c + a \quad \textit{(2)}
$$
$$
\frac{13}{c} = a + b \quad \textit{(3)}
$$

Our goal is to find $ a + b + c = \frac{m}{n} $,
with $ m, n $ relatively prime positive integers,
and then compute $ m+n $.

---

### Step 1: Set $ S = a + b + c $

From (1):
$ \frac{5}{a} = b + c $

But $ b + c = S - a $, so:

$$
\frac{5}{a} = S - a \quad \textit{(1')}
$$

Multiply both sides by $ a $:

$$
5 = a(S - a) = aS - a^2
\Rightarrow aS - a^2 = 5 \quad \textit{(A)}
$$

Similarly from (2):
$ \frac{10}{b} = c + a = S - b $

So:

$$
\frac{10}{b} = S - b \Rightarrow 10 = b(S - b) = bS - b^2 \quad \textit{(2')}
$$

$$
bS - b^2 = 10 \quad \textit{(B)}
$$

And from (3):
$ \frac{13}{c} = a + b = S - c $

So:

$$
\frac{13}{c} = S - c \Rightarrow 13 = c(S - c) = cS - c^2 \quad \textit{(3')}
$$

$$
cS - c^2 = 13 \quad \textit{(C)}
$$

So we now have:

- (A): $ aS - a^2 = 5 $
- (B): $ bS - b^2 = 10 $
- (C): $ cS - c^2 = 13 $

```

1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414

1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486

1487 We can solve each for a^2 , b^2 , c^2 or directly
1488 solve for a, b, c in terms of S , then sum them
1489
1490
1491
1492 ---
1493
1494 ### Step 2: Express a , b , c from each equation
1495
1496 From (A):
1497 \$\$
1498 $aS - a^2 = 5 \rightarrow a^2 - aS + 5 = 0$ \quad \text{wait: sign}
1499
1500
1501
1502 Wait: $aS - a^2 = 5 \rightarrow -a^2 + aS - 5 = 0$ \quad \text{Rightarrow } a^2 - aS + 5 = 0? \text{ No: signs reversed.}
1503
1504
1505 Wait:
1506
1507 Equation:
1508 $aS - a^2 = 5$ \$
1509
1510 Bring all to left:
1511
1512 $aS - a^2 - 5 = 0 \rightarrow -a^2 + aS - 5 = 0$ \$
1513
1514 Multiply both sides by -1 :
1515
1516 $a^2 - aS + 5 = 0$ \$? Wait:
1517
1518 No:
1519
1520 Negative times: $a^2 - aS + 5 = 0$ \$? Let's plug in.
1521
1522 Actually:
1523
1524 $-a^2 + Sa - 5 = 0$ \$
1525
1526 Multiply both sides by -1 :
1527
1528 $a^2 - Sa + 5 = 0$ \$
1529
1530 Wait, that would be $a^2 - Sa + 5 = 0$ \$, but
1531 original is $Sa - a^2 = 5$ \$
1532
1533 Check:
1534 Left side: $Sa - a^2$ \$, then $Sa - a^2 - 5 = 0$ \$
1535 $\rightarrow -a^2 + Sa - 5 = 0$ \$ \rightarrow multiply by -1 : $a^2 - Sa + 5 = 0$ \$
1536
1537 But that means $a^2 - Sa + 5 = 0$ \$
1538
1539 Wait, but if real roots: discriminant $S^2 - 4*1*5 = S^2 - 20$ \$
1540
1541 Can't be negative. Since $a > 0$ \$, so that's okay.
1542
1543 But go back - is this equation right?
1544
1545 Wait:
1546
1547 Equation (A): $Sa - a^2 = 5$ \$
1548
1549 Then $-a^2 + Sa - 5 = 0$ \$
1550
1551 Yes \rightarrow multiply both sides by -1 : $a^2 - Sa + 5 = 0$ \$
1552
1553 Wait - that's inconsistent: from $Sa - a^2 = 5$ \$,
1554 then $Sa - a^2 = 5$ \$.
1555 Then $-a^2 + Sa = 5$ \$
1556
1557 Then rearranged: $a^2 - Sa = -5$ \$? Yes.
1558
1559 That's better.
1560
1561 Wait, I think I made a sign error in the switch.
1562
1563 So instead:
1564
1565 From (A):
1566 $Sa - a^2 = 5$ \$
1567 $\rightarrow -a^2 + Sa - 5 = 0$ \$
1568
1569 Multiply through by -1 :
1570 $a^2 - Sa + 5 = 0$ \$? Wait \rightarrow no:
1571
1572 $-(a^2 + Sa - 5) = a^2 - Sa + 5$ \$? Wait: only if we negate the entire equation.
1573
1574 Yes.
1575
1576 If $-a^2 + Sa - 5 = 0$ \$, then multiplying by -1 :
1577 $a^2 - Sa + 5 = 0$ \$
1578
1579 Wait but original is positive. So if a solution

exists, check.
But for small S , maybe a is small.
Wait, better not to multiply - keep as:
From $Sa - a^2 = 5$ \$
We can solve quadratic:
 $a^2 - Sa + 5 = 0$ \$? Wait, no.
Wait:
 $-a^2 + Sa - 5 = 0$ \$
Multiply both sides by -1 :
 $a^2 - Sa + 5 = 0$ \$
Yes.
So quadratic:
 $a^2 - Sa + 5 = 0$ \quad \text{(A')}\$
Wait - if this is correct, then roots of $a^2 - Sa + 5 = 0$ \$
But this means $a = \frac{S \pm \sqrt{S^2 - 20}}{2}$ \$
Similarly, for (B): $bS - b^2 = 10$ \$
So $-b^2 + bS - 10 = 0$ \$
 \rightarrow Multiply by -1 : $b^2 - bS + 10 = 0$ \$
 $\rightarrow b = \frac{S \pm \sqrt{S^2 - 40}}{2}$ \$
Similarly (3): $cS - c^2 = 13$ \$
 $\rightarrow c^2 - cS + 13 = 0$ \$
Wait, let's check:
From $cS - c^2 = 13$ \$
 $\rightarrow -c^2 + cS = 13$ \$
 $\rightarrow c^2 - cS + 13 = 0$ \$? No:
Again: move things:
 $-c^2 + cS - 13 = 0$ \$
 \rightarrow multiply by -1 : $c^2 - cS + 13 = 0$ \$
Wait same process.
Wait \rightarrow from $cS - c^2 = 13$ \$
 $\rightarrow -c^2 + cS - 13 = 0$ \$
Multiply by -1 : $c^2 - cS + 13 = 0$ \$
Wait \rightarrow so yes:
We get quadratics:
 $a^2 - Sa + 5 = 0$ \$ \rightarrow (A')
 $b^2 - bS + 10 = 0$ \$ \rightarrow (B')
 $c^2 - cS + 13 = 0$ \$ \rightarrow (C')
Wait - but that suggests a, b, c satisfy such quadratics, but we also know $a + b + c = S$ \$
So from each:
We solve for a, b, c \$:
 $a = \frac{S \pm \sqrt{S^2 - 20}}{2}$ \$
 $b = \frac{S \pm \sqrt{S^2 - 40}}{2}$ \$
 $c = \frac{S \pm \sqrt{S^2 - 52}}{2}$ \$
Wait, because quadratic:
 $x^2 - Sx + k = 0 \rightarrow$ solutions $x = \frac{S \pm \sqrt{S^2 - 4k}}{2}$ \$
Wait, in (A): equation: $a^2 - Sa + 5 = 0$ \$ \rightarrow so discriminant $S^2 - 4*1*5 = S^2 - 20$ \$, so roots:
 $a = \frac{S \pm \sqrt{S^2 - 20}}{2}$ \$
Similarly, for (B): $S^2 - 40$ \$

1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680

1681 c: $S^2 - 52$

1682

1683 But since $a, b, c > 0$, we must pick the correct

1684 sign.

1685

1686 Let us see: Assume that values are less than S (

1687 since they are positive). Since $a + b + c = S$,

1688 and all positive, each must be less than S .

1689

1690 Now, the roots of $x^2 - Sx + k=0$:

1691

1692 The sum is S , so average is $S/2$. For each

1693 pair.

1694

1695 So the roots are symmetric about $S/2$.

1696

1697 But we have $a = \frac{S + \sqrt{S^2 - 20}}{2}$, with d_a

1698 $= \sqrt{S^2 - 20}$

1699

1700 Let's calculate both roots:

1701

1702 $a_+ = \frac{S + \sqrt{S^2 - 20}}{2}$

1703

1704 $a_- = \frac{S - \sqrt{S^2 - 20}}{2}$

1705

1706 Now, since $a < S$, and both roots are less than S

1707 ? First, $a_+ < S$? Yes.

1708

1709 But is $a_+ > S/2$? Yes.

1710

1711 We need $a + b + c = S$. So now suppose we pick

1712 all roots equal to the minus version? Unlikely.

1713

1714 But since $a, b, c > 0$, and sum to S , we need

1715 $a, b, c < S$.

1716

1717 Now, the two roots: one is $S/2 + d/2$, one $S/2$

1718 $- d/2$

1719

1720 Since $d = \sqrt{S^2 - 4k}$, and $k > 0$, small.

1721

1722 But if we take the larger root: for a , $a_+ > S/2$

1723

1724 Likewise, $a_- < S/2$

1725

1726 Similarly for others.

1727

1728 But if we take the larger root for all three, their

1729 total sum can be?

1730

1731 Wait: suppose we take:

1732

1733 $a: \frac{S + \sqrt{S^2 - 20}}{2}$

1734

1735 $b: \frac{S + \sqrt{S^2 - 40}}{2}$

1736

1737 $c: \frac{S + \sqrt{S^2 - 52}}{2}$

1738

1739 Then sum:

1740

1741 $\frac{S}{2} * 3 + \frac{1}{2}(\sqrt{S^2 - 20} + \sqrt{S^2 - 40}$

1742 $+ \sqrt{S^2 - 52}) = S$

1743

1744 $= \frac{3S}{2} + \frac{1}{2}(\dots)$

1745

1746 But this would be $> \frac{3S}{2} > S$ unless S

1747 < 0 - not possible.

1748

1749 So impossible. So cannot pick all as larger root.

1750

1751 Similarly, if pick all as smaller roots:

1752

1753 $a_- = \frac{S - \sqrt{S^2 - 20}}{2}$

1754

1755 Then sum:

1756

1757 $\frac{3S}{2} - \frac{1}{2}(\sqrt{S^2 - 20} + \sqrt{S^2 - 40}$

1758 $+ \sqrt{S^2 - 52}) = S$

1759

1760 This sum is likely less than S , since the

1761 subtracted term is positive.

1762

1763 So we need to pick two big and one small, or other

1764 combinations.

1765

1766 But maybe none of them work. Perhaps only one of

1767 them is chosen.

1768

1769 Since a, b, c are symmetric in sum, maybe the

1770 values are such that the terms are picked

1771 consistently.

1772

1773 Alternative idea: add the three equations:

1774

1775 We have:

1776

1777

From (1'), (2'), (3'):

We can write equations:

$aS - a^2 = 5$

$bS - b^2 = 10$

$cS - c^2 = 13$

Add them together:

$S(a + b + c) - (a^2 + b^2 + c^2) = 5 + 10 + 13 = 28$

But $a + b + c = S$, so first term: $S \cdot S = S^2$

So:

$S^2 - (a^2 + b^2 + c^2) = 28$ (D)

Also, we can find $a^2 + b^2 + c^2$ in terms of $(a+b+c)^2$

$(a + b + c)^2 = a^2 + b^2 + c^2 + 2(ab + bc + ca)$

So:

$S^2 = (a^2 + b^2 + c^2) + 2(ab+bc+ca)$

From (D):

$a^2 + b^2 + c^2 = S^2 - 28$

Plug in:

$S^2 = (S^2 - 28) + 2(ab + bc + ca)$

Then:

$-28 = -2(ab+bc+ca)$

So $ab + bc + ca = 14$

Nice.

So now we know:

$S = a+b+c$

$ab + bc + ca = 14$

$a^2 + b^2 + c^2 = S^2 - 28$

Now go back to the equations.

From earlier, we have expressions for $b + c = \frac{5}{a}$, etc.

From (1): $b + c = \frac{5}{a}$

But $b + c = S - a$

So again $S - a = \frac{5}{a} \rightarrow a(S - a) = 5$

Same as before: $aS - a^2 = 5$

Since $(b + c) = S - a = 5/a$

Similarly, $S - b = 10/b$, etc.

Now, we already now that $ab + bc + ca = 14$

Now, we can use:

We know $a(S - a) = 5$

Similarly $b(S - b) = 10$, $c(S - c) = 13$

Wait that's more useful.

Let's write:

$a(S - a) = 5$

$\rightarrow aS - a^2 = 5$

But also we know $ab + bc + ca = 14$

Now, perhaps we can express $ab + bc + ca$ using these expressions.

1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874

1875 We know from above:
1876
1877 We also know from equations that:
1878
1879 Since $b + c = S - a$, we can write $ab + ac = a(b+c) = a(S - a) = 5$
1880
1881 Wait! Spot:
1882
1883 From $b + c = S - a$, multiply by a :
1884
1885 $a(b + c) = a(S - a)$
1886
1887 But left side is $ab + ac$, so:
1888
1889 $ab + ac = a(S - a) = 5$
1890
1891 Similarly, by symmetry:
1892
1893 From equation (2), $c + a = 10 / b$
1894
1895 Then, multiply both sides by b :
1896
1897 $b(c + a) = 10$, so $bc + ab = 10$
1898
1899 Wait - so:
1900
1901 $ab + bc = 10$
1902
1903 Wait no:
1904
1905 Wait: $c+a = S - b$, and from (2), $S - b = 10 / b$ -> so:
1906
1907 $b(c + a) = 10$
1908
1909 But $b(c + a) = bc + ab$
1910
1911 Thus:
1912
1913 $ab + bc = 10$
1914
1915 Similarly, equation (3): $a + b = 13 / c$, so $c(a+b) = 13$
1916
1917 So $ca + cb = 13$
1918
1919 That is:
1920
1921 $ca + cb = 13$
1922
1923 So now, we have:
1924
1925 From first: $ab + ac = 5$
1926
1927 From second: $ab + bc = 10$
1928
1929 From third: $bc + ca = 14$? Wait no.
1930
1931 Wait solve carefully:
1932
1933 Wait first: $ab + ac = 5$
1934
1935 Second: $ab + bc = 10$
1936
1937 Third: from $c(a+b) = 13$: $c(a+b) = ca + cb = 13$
1938
1939 So indeed:
1940
1941 $- ab + ac = 5$ -> (i)
1942
1943 $- ab + bc = 10$ -> (ii)
1944
1945 $- bc + ca = 13$ -> (iii)
1946
1947 But wait - we previously found $ab+bc+ca = 14$
1948
1949 Wait now we have:
1950
1951 Add all three:
1952
1953 (i)+(ii)+(iii):
1954
1955 Left: $ab + ac + ab + bc + bc + ca$
1956
1957 $= 2ab + 2bc + 2ca$
1958
1959 So:
1960
1961 $2(ab + bc + ca) = 5+10+13=28$
1962
1963 So $2(ab+bc+ca)=28$ -> $ab+bc+ca = 14$ -> matches what we got earlier.
1964
1965 Good - consistent.
1966
1967 Now, let's label:

Let:
 $- x = ab$
 $- y = bc$
 $- z = ca$
Then from above:
 $- x + z = 5$ -> (from $ab+ac=ab+ac=x+z=5$)
 $- x + y = 10$
 $- y + z = 13$
And $x+y+z = 14$
Now solve this system.
From:
(1) $x + z = 5$
(2) $x + y = 10$
(3) $y + z = 13$
Subtract (1) from (2): (2)-(1):
 $(x + y) - (x+z) = 10 - 5 \Rightarrow y - z = 5$
From (3): $y + z = 13$
Now solve:
From (3) and $(y-z=5)$:
Add them:
 $(y+z)+(y-z)=13+5=18 \Rightarrow 2y=18 \Rightarrow y=9$
Then from $y + z = 13$, $z = 13 - 9 = 4$
From (2): $x + y = 10 \Rightarrow x = 10 - 9 = 1$
From (1): $x + z = 1 + 4 = 5$: yes, checks.
So:
 $- ab = 1$
 $- bc = 9$
 $- ca = 4$
Nice.
So now we know:
 $- ab = 1$
 $- bc = 9$
 $- ca = 4$
Now we can find a, b, c from these?
We know:
From $ab=1 \Rightarrow b = \frac{1}{a}$
From $ca=4 \Rightarrow c = \frac{4}{a}$
Now plug into another: $bc = 9$
 $b = \frac{1}{a}$, $c = \frac{4}{a}$
So $b \cdot c = \frac{1}{a} \cdot \frac{4}{a} = \frac{4}{a^2}$
We know this should be 9.
So:
 $\frac{4}{a^2} = 9$
 $\Rightarrow a^2 = \frac{4}{9}$
 $\Rightarrow a = \frac{2}{3}$ (since $a > 0$)
Now compute b and c :
From $ab = 1 \Rightarrow b = \frac{1}{a} = \frac{3}{2}$
From $ca = 4 \Rightarrow c = \frac{4}{a} = 4 \cdot \frac{3}{2} = 6$
Wait: $a = 2/3$, so $c = 4 / (2/3) = 4 \cdot 3 / 2 = 6$
Yes.
Check bc : $b = 3/2$, $c = 6$, so $bc = 3/2 \cdot 6 = 9$: yes.

1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068

```

2069 Check $ ab = (2/3)(3/2)=1 $: yes.
2070
2071 Now we have $ a = \frac{2}{3} $, $ b = \frac{3}{2} $
2072 , $ c=6 $
2073
2074 Now compute $ S = a + b + c $
2075
2076 $$
2077 S = \frac{2}{3} + \frac{3}{2} + 6
2078 $$
2079
2080 We compute this:
2081
2082 Common denominator: 6
2083
2084 $ \frac{2}{3} = \frac{4}{6} $
2085 $ \frac{3}{2} = \frac{9}{6} $
2086 $ 6 = \frac{36}{6} $
2087
2088 Sum: $ \frac{4+9+36}{6} = \frac{49}{6} $
2089
2090 So $ S = \frac{49}{6} $
2091
2092 Now - check if this satisfies original equations.
2093
2094 ### Check equation (1): $ \frac{5}{a} = b + c $?
2095 $ a = 2/3 $ -> $ 5 / (2/3) = 5 * 3 / 2 = 15/2 = 7.5 $
2096
2097 Now $ b + c = 3/2 + 6 = 3/2 + 12/2 = 15/2 $: yes ->
2098 7.5 -> correct
2099
2100 ### Equation (2): $ \frac{10}{b} = c + a $?
2101 $ b = 3/2 $: $ 10 / (3/2) = 10 \cdot 2/3 = 20/3 \approx
2102 6.666 $
2103
2104 $ c + a = 6 + 2/3 = 18/3 + 2/3 = 20/3 $ -> yes
2105
2106 ### Equation (3): $ \frac{13}{c} = a + b $?
2107 $ c=6 $: $ 13 / 6 \approx 2.166 $
2108
2109 $ a + b = 2/3 + 3/2 = 4/6 + 9/6 = 13/6 $ -> equals $
2110 13/6 $: yes
2111
2112 Perfect -> all match.
2113
2114 Thus, $ a + b + c = \frac{49}{6} $
2115
2116 Now $ m = 49 $, $ n=6 $: relatively prime?
2117 Yes, gcd(49,6)=1 since 49=7^2, 6=2 \times 3 -> no
2118 common factors.
2119
2120 Thus $ m+n=49+6 = \boxed{55} $
2121
2122 ### Final Answer:
2123
2124 $ \boxed{55} $

```
