

O1-Pruner: Length-Harmonizing Fine-Tuning for O1-Like Reasoning Pruning

Haotian Luo¹, Haiying He², Yibo Wang³, Shiwei Liu⁴,
Wei Li⁵, Xiaochun Cao¹, Dacheng Tao⁶, Naiqiang Tan⁵, Li Shen^{1,*},

¹Shenzhen Campus of Sun Yat-sen University, ²China Agriculture University, ³Tsinghua University,
⁴University of Oxford, ⁵Didichuxing Co. Ltd, ⁶Nanyang Technological University

Correspondence: [mathshenli@gmail.com]

Abstract

Recently, long-thought reasoning LLMs, such as OpenAI’s O1, adopt extended reasoning processes similar to how humans ponder over complex problems. This reasoning paradigm significantly enhances the model’s problem-solving abilities and achieves promising results. However, long-thought reasoning process leads to a substantial increase in inference time. A pressing challenge is reducing the inference overhead of long-thought LLMs while ensuring accuracy. In this paper, we identify that long-thought reasoning models struggle to effectively allocate token budgets based on problem difficulty and reasoning redundancies. To address this, we propose Length-Harmonizing Fine-Tuning (O1-Pruner), aiming at minimizing reasoning overhead while maintaining accuracy. This effective fine-tuning method first estimates the LLM’s baseline performance through pre-sampling and then uses RL-style fine-tuning to encourage the model to generate shorter reasoning processes under accuracy constraints. This allows the model to achieve efficient reasoning with lower redundancy while maintaining accuracy. Experiments on various mathematical reasoning benchmarks show that O1-Pruner not only significantly reduces inference overhead but also achieves higher accuracy, providing a novel and promising solution to this challenge.

1 Introduction

Reasoning represents a fundamental capability of large language models (LLMs), serving as a cornerstone in the advancement of artificial intelligence research (Huang and Chang, 2023). Recently OpenAI’s O1 (OpenAI, 2024) have introduced long-thought reasoning models that mimic human-like problem-solving processes. In addition to O1, researchers have also developed models that inference with a similar long-thought reasoning pattern, such as Deepseek-R1 (DeepSeek,

2024), QwQ (Qwen, 2024) and Marco-o1 (Zhao et al., 2024). These models leverage a long chain-of-thought framework, enabling them to tackle complex problems by iteratively identifying and correcting errors, simplifying intricate steps, and exploring alternative strategies when initial approaches prove inadequate. Furthermore, Mulberry (Yao et al., 2024) has demonstrated that O1-Like reasoning can also play a significant role in multimodal reasoning. This reasoning paradigm significantly enhances the problem-solving capabilities of large language models (LLMs) by allowing them to approach complex tasks in a more systematic and human-like manner, demonstrating an ability to handle problems that would otherwise be challenging or intractable for conventional LLMs.

While long-thought reasoning enhances reasoning capabilities and improves accuracy, it is accompanied by longer output sequences, which result in increased computational overhead. A critical challenge lies in developing mechanisms that enable LLMs to dynamically adjust the length and complexity of their reasoning processes in accordance with the difficulty of the problems they encounter.

In this paper, we first revisit the long-thought reasoning processes. we observe that the reasoning processes in long-thought reasoning LLMs often exhibit significant redundancies, which leads to inefficient use of computational resources. This inefficiency not only increases inference costs but also highlights a fundamental limitation in the models’ ability to adapt their reasoning depth to suit the demands of diverse tasks. Building on this analysis, we formulate an optimization objective aimed at minimizing reasoning overhead while maintaining accuracy as a constraint. Our approach introduces a Length-Harmonizing Reward, which explicitly rewards shorter solutions while penalizing accuracy degradation. By embedding this reward into a RL-based framework, we enable

the model to optimize for efficiency without compromising performance. Moreover, our method incorporates an off-policy training strategy inspired by Proximal Policy Optimization (PPO), which aimed at reducing training complexity while maintaining robustness.

Our experiments are conducted using open-source long-thought reasoning LLMs, and we compare our approach against several competing methods like SFT and DPO (Rafailov et al., 2024). Through extensive experiments, we demonstrate the efficiency of our proposed methods. Additionally, we perform further studies on the influence of hyperparameters and dataset difficulty on our approach, in order to gain deeper insights into the characteristics and behavior of this novel framework.

In conclusion, our contributions can be outlined as follows:

- We design a simple experiment and identify a critical issue in the reasoning process of long-thought models, referred to as length disharmony, which leads to redundant inference overhead.
- We formulate an optimization problem aimed at improving model inference efficiency while maintaining accuracy, and based on this, we propose Length-Harmonizing Fine-Tuning (**O1-Pruner**) approach.
- Through extensive experiments, we demonstrate the effectiveness of **O1-Pruner** and conduct in-depth further evaluation, to provide insights and inspiration for future research in this area.

2 Related Work

Inference-time Scaling. Inference-time scaling refers to the ability of large language models (LLMs) to improve their outputs by utilizing additional computation during inference time. Recent studies (Snell et al., 2024; Wu et al., 2024) have explored how scaling inference-time computation can enhance the performance of LLMs on challenging prompts. This approach draws parallels to human reasoning, where additional cognitive effort is often allocated to complex problems. In addition to increasing the number of candidate solutions or searching different steps, OpenAI’s O1 inference (OpenAI, 2024) demonstrates that extending the length of the solution generated during rea-

soning can also significantly enhance the model’s performance.

CoT Compression. Long-CoT introduces much more computational overhead. To address this, recent studies propose various compression strategies. For instance, token-budget-aware frameworks (Han et al., 2024a) dynamically allocate tokens based on task complexity. C3oT (Kang et al., 2024) uses GPT-4 to compress CoT while preserving key information. Continuous representation methods like CCoT (Cheng and Durme, 2024) and COCONUT (Hao et al., 2024) reduce redundancy by reasoning in latent spaces with fewer or variable-length tokens. Unlike these methods that focus on step-level compression, our approach targets long-form reasoning by optimizing reasoning paths instead of compressing individual steps. Concurrently, works like (Chen et al., 2024) tackle overthinking using preference-based optimization (SimPO (Meng et al., 2024)), and (Team et al., 2025) apply long2short RL to enhance short-CoT models. In contrast, we examine long reasoning from the perspective of length distribution and formulate an RL-based optimization framework with accuracy concern, offering a novel perspective for future research.

3 Revisiting the “Length Disharmony”

We define “Length Disharmony” as the inefficiency in long-thought reasoning where shorter responses achieve high accuracy, making longer ones unnecessary and wasteful—especially given the Transformers quadratic complexity, which increases inference time. To illustrate this, we conducted a simple experiment: 64 problems were randomly selected from the MATH test set (excluding hard ones for QwQ-32B). Using Top-P sampling, we generated 512 solutions per problem with both Marco-o1 and QwQ-32B. The solutions were grouped into four length-based intervals, and accuracy was computed for each, revealing the disharmony in solution lengths.

Accuracy-Length Relationship at Instance Level. From the data we collected, we can ascertain the relationship between accuracy and length at the instance level, which is shown in Figure 1. It is evident that there exists a markedly inconsistent relationship between length and accuracy across different problems. The highest accuracy may manifest within the shortest, intermediate, or longest length intervals. Specifically, we

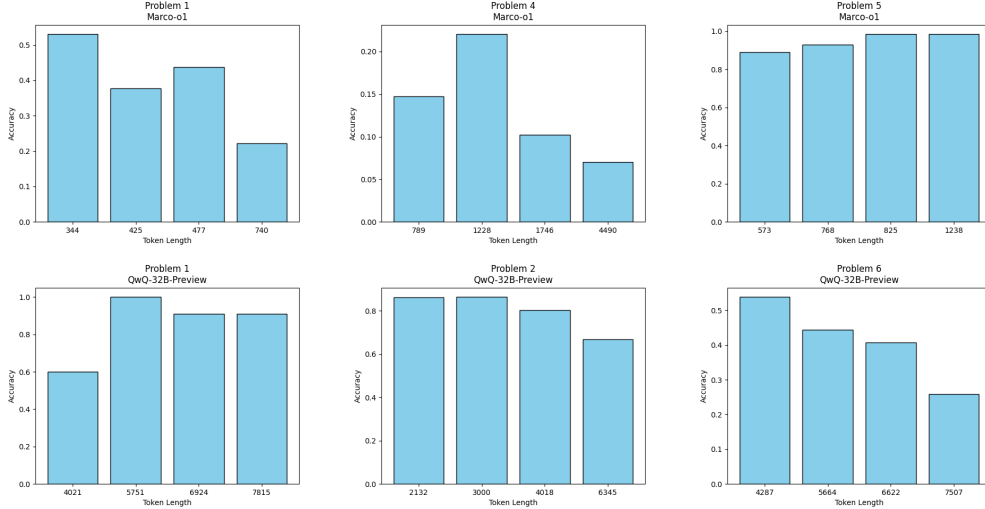


Figure 1: Accuracy-Length Relationship at Instance level. The relationship between length and accuracy varies significantly across problems, with peak accuracy occurring at short, medium, or long intervals. Notably, high accuracy often persists in shorter intervals.

observe that relatively high accuracy is preserved even within shorter-length intervals.

Accuracy-Length Relationship at Distribution Level. Furthermore, by calculating the average accuracy across all problems within different intervals, we have derived the relationship between accuracy and length at the distribution level, which is shown in Table 1. At the distribution level, our analysis reveals a consistent trend where shorter response lengths are associated with higher average accuracy rates. This observation can be explained by the premise that a shorter response length typically signifies the model’s ability to identify the optimal solution path more efficiently, consequently requiring fewer iterative processes of reflection and backtracking.

Therefore, we can conclude that long-thought models exhibit a phenomenon of length disharmony during reasoning, which leads to redundant computational overhead in the inference phase. This reasoning redundancy can be mitigated, as high accuracy is still maintained even at shorter lengths. From this perspective, we propose Length-Harmonizing Fine-Tuning (O1-Pruner) to optimize long-thought reasoning, enabling it to maintain high accuracy while reducing inference redundancy.

Table 1: Accuracy-Length Relationship at Distribution Level. A larger interval number indicates a longer solution length. The average accuracy is higher when the solution length is short.

Model	Interval 1	Interval 2	Interval 3	Interval 4
Marco	81.1	80.2	78.8	75.3
QwQ	44.9	49.9	45.9	45.3

4 Methodology

4.1 Problem Setup

We consider a LLM parameterized by θ and denoted as π_θ . In the context of math problem solving, the LLM accepts a sequence $\mathbf{x} = [x_1, \dots, x_n]$, commonly termed as the problem, and then generate a corresponding solution $\mathbf{y} = [y_1, \dots, y_m]$. Hence, the solution \mathbf{y} is construed as a sample drawn from the conditional probability distribution $\pi_\theta(\cdot|\mathbf{x})$. The conditional probability distribution $\pi_\theta(\mathbf{y}|\mathbf{x})$ can be decomposed as follows:

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^m \pi_\theta(y_j|\mathbf{x}, \mathbf{y}_{<j}). \quad (1)$$

Firstly, we review the process of supervised fine-tuning (SFT). SFT is the primary method to adapt a pre-trained LLM for downstream tasks with a relatively smaller supervised dataset of labeled examples compared to the data of pre-training stage. In this paper, we focus on the task of mathematic problem solving where the problem-solution pairs denoted as (\mathbf{x}, \mathbf{y}) , are drawn from a specified SFT dataset \mathcal{D} . Thus the training objective of SFT under this setting can be formulated as:

$$\max_{\pi_\theta} \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\log \pi_\theta(\mathbf{y} | \mathbf{x}) \right]. \quad (2)$$

4.2 O1-Pruner

To start with, let’s assume that π_θ is a LLM that can solve math problems with long thought with redundancy and disharmony. we hypothesize that the reasoning paths represented by output thought

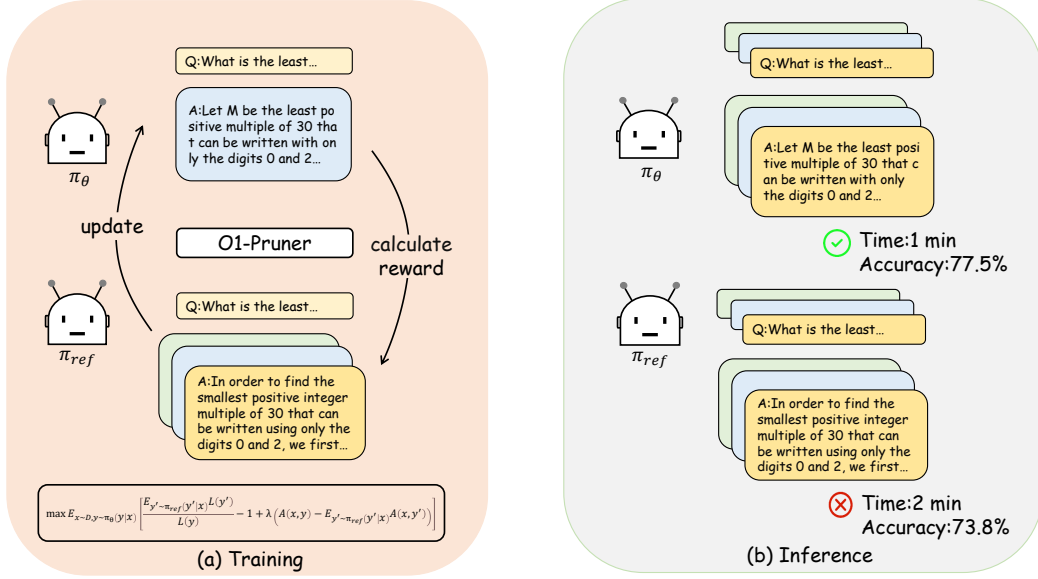


Figure 2: Length-Harmonizing Fine-Tuning. During the training phase, for each problem, we sample multiple times from the reference model. Subsequently, we sample from the model to be optimized and compute the reward based on the reference samples, followed by a RL-style fine-tuning. During the inference phase, the model optimized through **O1-Pruner** demonstrates a significant improvement in inference speed, along with a noticeable enhancement in accuracy.

of language model π_θ contain redundancies and lack proper coordination. To address this, we propose an optimization objective that ensures no degradation in accuracy while tackling the issue from two perspectives. First, at the overall level, we aim to shorten the reasoning paths. Second, we encourage the model to output shorter answers for simpler problems, while for more complex problems, we guide the model to learn the correct reasoning paths, which, according to the inference scaling law, typically involve longer reasoning sequences. Given a problem x , we define $L(y)$ as the length (counted by token) of the solution y . Considering a reference model π_{ref} , we reduce the solution length of the policy model relative to that of the reference model, which can be formulated as:

$$\max \mathbb{E}_{x \sim D} \left[\mathbb{E}_{y \sim \pi_\theta(y|x), y' \sim \pi_{ref}(y|x)} \frac{L(y')}{L(y)} - 1 \right]. \quad (3)$$

We subtract a constant 1 from the optimization objective to ensure that the initial expected value of the optimization is zero. We then define an accuracy function $A(x, y, \text{answer})$, which takes the problem, solution, and the real answer as inputs, and returns 0 or 1 to indicate whether the solution is incorrect or correct. For the sake of simplicity in the notation, we omit the real answer, denoting the function as $A(x, y)$. We aim to ensure that the

model's accuracy does not decrease, or even improves, during the process of optimizing for length. Thus, we derive the following constraint condition Eq.4. Therefore, we can establish our optimization objective as Eq.5.

To solve this constrained optimization problem, we incorporate constraint into the objective function as a penalty term. Specifically, the constraint on accuracy is added to the objective with a penalty weight $\lambda \geq 0$:

$$\max \mathbb{E}_{x \sim D, y \sim \pi_\theta(y|x), y' \sim \pi_{ref}(y|x)} \frac{L(y')}{L(y)} - 1 + \lambda(A(x, y) - A(x, y')). \quad (7)$$

By reorganizing the terms related with reference model π_{ref} , we have:

$$\max \mathbb{E}_{x \sim D, y \sim \pi_\theta(y|x)} \frac{\mathbb{E}_{y' \sim \pi_{ref}(y'|x)} L(y')}{L(y)} - 1 + \lambda(A(x, y) - \mathbb{E}_{y' \sim \pi_{ref}(y'|x)} A(x, y')). \quad (8)$$

In practice, we approximate the expectation terms related with π_{ref} by sampling. For each x , we sample for K times from $\pi_{ref}(\cdot|x)$ and calculate

$$\mathbb{E}_{x \sim D, y \sim \pi_\theta(y|x)} A(x, y) \geq \mathbb{E}_{x \sim D, y' \sim \pi_{ref}(y'|x)} A(x, y') \quad (4)$$

$$\max \mathbb{E}_{x \sim D} \left[\mathbb{E}_{y \sim \pi_\theta(y|x), y' \sim \pi_{ref}(y'|x)} \frac{L(y')}{L(y)} - 1 \right] \text{ s.t. } \mathbb{E}_{x \sim D, y \sim \pi_\theta(y|x)} A(x, y) \geq \mathbb{E}_{x \sim D, y' \sim \pi_{ref}(y'|x)} A(x, y') \quad (5)$$

$$L^{\text{LH}}(\theta; x, y) = -\mathbb{E}_{x \sim D, y \sim \pi_{ref}(y|x)} \left[\min(r(\theta) R_{LH}(x, y), \text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon) R_{LH}(x, y)) \right] \quad (6)$$

the mean value:

$$\bar{L}_{ref}(x) = \frac{1}{K} \sum_{i=1}^K L(y'_i), \quad y'_i \sim \pi_{ref}(\cdot | x); \quad (9)$$

$$\bar{A}_{ref}(x) = \frac{1}{K} \sum_{i=1}^K A(x, y'_i), \quad y'_i \sim \pi_{ref}(\cdot | x); \quad (10)$$

This approach is widely employed in Policy Gradient with Baseline. Furthermore, a recently proposed method GRPO (Shao et al., 2024) adopts a similar technique to reduce training overhead. Based on this technique, our objective can be approximated as:

$$\max \mathbb{E}_{x \sim D, y \sim \pi_\theta(y|x)} \frac{\bar{L}_{ref}(x)}{L(y)} - 1 + \lambda(A(x, y) - \bar{A}_{ref}(x)). \quad (11)$$

Since both $L(y)$ and $A(x, y)$ are not differentiable, we solving this optimization with policy gradient approach, which is shown to have strong performance despite its simplicity. Furthermore, it is worth noting that during the optimization process, frequent sampling from the current distribution π_θ is required during training, which significantly increases the complexity of the training procedure. Considering that off-policy training can bring remarkable effectiveness with pre-collected data, we adopt an off-policy training approach by directly sampling from the π_{ref} instead of π_θ . Besides, since our reward is derived by assessing the merit of a sample within the distribution relative to the expected outcome, our reward can be regarded as an approximate advantage function. Consequently, we employ a PPO-style loss (Schulman et al., 2017) to optimize the objective function, which helps for our off-policy training

strategy. Defining the Length-Harmonizing Reward $R_{LH}(x, y) = \frac{\bar{L}_{ref}(x)}{L(y)} - 1 + \lambda(A(x, y) - \bar{A}_{ref}(x))$, the loss function of off-policy-version Length-Harmonizing Fine-Tuning is Eq.6, where $r(\theta) = \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)}$. $\text{clip}()$ is the clipping function.

This allows us to prepare the required data at the beginning of training, thereby greatly simplifying the training workflow. Our experiments show that this off-policy approach still enables our method to achieve outstanding performance, significantly surpassing other baselines.

4.3 Understanding the Loss Function

To intuitively understand how our loss function works, we begin by analyzing the R_{LH} term. Evidently, R_{LH} comprises two distinct components, namely the length reward term $\frac{\bar{L}(x, \pi_{ref})}{L(y)} - 1$ and the accuracy reward term $\lambda(A(x, y) - \bar{A}(x, \pi_{ref}))$. Obviously, the length reward term will reward shorter outputs. When the sequence length are consistent with expected output length of reference model, the length reward is 0; however, when the output is longer, the length reward becomes negative. The accuracy reward term is essential for balancing length and accuracy. For a problem x with a relatively high accuracy expectation, solving it correctly does not yield a significant accuracy reward. As a result, the model tends to explore shorter solutions. For more challenging problems, solving them correctly yields a higher accuracy reward, indicating that we do not want the model to prioritize shortening the output. Instead, we aim for the model to focus on generating a correct solution. On this basis, if the correct solution is relatively short, the model will receive an additional length reward.

To the end, we summarize the training procedure of our proposed O1-Pruner in Algorithm 1.

Algorithm 1 O1-Pruner

1: **Input:** LLM π_θ , Dataset $\mathcal{D} = \{(x^i, a^i)\}_{i \in [N]}$

2: **Initialize:** $\pi_{ref} = \pi_\theta$

3: **for** $i = 1$ **to** N **do**

4: sampling K solutions y_1^i, \dots, y_K^i from $\pi_{ref}(\cdot | x_i)$

5: calculating $\bar{L}_{ref}(x^i) = \frac{1}{K} \sum_{k=1}^K L(y_k^i)$

6: calculating $\bar{A}_{ref}(x^i) = \frac{1}{K} \sum_{k=1}^K A(x^i, y_k^i)$

7: randomly select m ($m \leq K$) solutions from y_1^i, \dots, y_K^i

8: Update $\theta = \arg \min_{\theta} \sum_{j=1}^m L^{LH}(\theta; x^i, y_j^i)$

9: **end for**

10: **Output:** Updated LLM π_θ

5 Experiments

5.1 Experiment Setup

We fine-tune two strong models Marco-o1-7B with full-parameter tuning and QwQ-32B-Preview using Freeze Fine-Tune due to resource constraints. Training used 5k problem-answer pairs from MATH, generating 16 and 12 solutions per problem respectively. Evaluation spans MATH, GSM8k, and AIME25. Additional results on DeepSeek-R1-Distill-Qwen-7B are in the appendix.

5.2 Baselines.

To validate the superiority of our method for long thought reasoning optimization tasks, we have selected the three representatively comparative methods. **(i) Fast-Solving Prompt:** The Fast-Solving Prompt is a prompting technique wherein we instruct the model within the prompt to solve the given problem as swiftly as possible, aiming to achieve the desired reduction in reasoning length. **(ii) SFT:** For the SFT method, we curated the training dataset by selecting the two shortest correct solutions for each problem, ensuring that the model is exposed to examples that embody both accuracy and conciseness. These solutions were then used to train the model following the standard SFT pipeline. **(iii) DPO:** For the implementation of DPO, we meticulously selected two of the shortest correct solutions to serve as the chosen samples, which exemplify efficiency and precision in problem-solving. Conversely, to represent the reject sample, we opted for the longest solution

available.

Evaluation Metric. We employ the following average accuracy, average length and Accuracy-Efficiency Score (AES) as key metrics to assess whether the model achieves a desirable balance between reasoning accuracy and length:

- **Accuracy** Accuracy reflects whether the model correctly solves the problem. It is measured as the proportion of problems for which the model’s generated solution is correct. A higher accuracy indicates better problem-solving capability.
- **Length** Length denotes the number of tokens in the generated solution. It serves as a proxy for the computational cost of generating solutions, where a shorter length implies greater efficiency.
- **AES** We define a novel metric called Accuracy-Efficiency Score (AES), to evaluate the trade off between improving accuracy and reducing computational cost. It is calculated by weighting and summing the model’s solution length and accuracy. Defining $\Delta \text{Length} = \frac{\text{Length}_{\text{baseline}} - \text{Length}_{\text{model}}}{\text{Length}_{\text{baseline}}}$ and $\Delta \text{Acc} = \frac{\text{Acc}_{\text{model}} - \text{Acc}_{\text{baseline}}}{\text{Acc}_{\text{baseline}}}$, the AES is calculated by:

$$\text{AES} = \begin{cases} \alpha \cdot \Delta \text{Length} + \beta \cdot |\Delta \text{Acc}|, & \text{if } \Delta \text{Acc} \geq 0 \\ \alpha \cdot \Delta \text{Length} - \gamma \cdot |\Delta \text{Acc}|, & \text{if } \Delta \text{Acc} < 0 \end{cases}$$

where $\alpha > 0$, $\beta > 0$, and $\gamma > 0$. AES evaluates the trade-off between improving accuracy and reducing computational cost. And we emphasize the penalization of accuracy degradation by setting $\gamma > \beta$. We set the default values as $\alpha = 1$, $\beta = 3$, $\gamma = 5$.

5.3 Experimental Results

Table 2 demonstrates the performance of various methods across different evaluation metrics. The proposed **O1-Pruner** consistently achieves superior performance in balancing reasoning accuracy and efficiency compared to baseline and competing methods. Notably, it exhibits the best trade-off between accuracy and reasoning length across all datasets, as further supported by its significantly higher Accuracy-Efficiency Score (AES) values.

Table 2: Main Experiment Results. We present the performance of two selected models optimized through different methods across three mathematical reasoning datasets. It can be observed that the models trained with **O1-Pruner** achieve the best trade off between accuracy and length in comparison with other approaches.

Model	MATH			GSM8K			AIME25			AVERAGE		
	Acc	Length	AES	Acc	Length	AES	Acc	Length	AES	Acc	Length	AES
Marco-o1-7B												
<i>(full fine-tune)</i>												
Baseline	<u>73.8</u>	1156	0	89.2	530	0	7.3	1954	0	56.8	1213	0
Fast-solving Prompt	71.0	1113	-0.15	81.7	447	-0.26	5.3	1970	-1.38	52.7	1177	-0.60
SFT	73.6	1076	0.06	<u>89.9</u>	497	0.09	<u>6.7</u>	1652	<u>-0.26</u>	<u>56.7</u>	1075	<u>-0.04</u>
DPO	71.8	<u>761</u>	<u>0.21</u>	88.6	<u>410</u>	<u>0.19</u>	3.0	<u>1054</u>	-2.48	54.5	<u>742</u>	-0.69
O1-Pruner	<u>77.5</u>	<u>657</u>	<u>0.58</u>	<u>91.4</u>	<u>343</u>	<u>0.43</u>	<u>8.0</u>	<u>890</u>	<u>0.83</u>	<u>59.0</u>	<u>630</u>	<u>0.61</u>
QwQ-32B-Preview												
<i>(freeze fine-tune last 48 layers)</i>												
Baseline	90.6	2191	0	95.1	777	0	28.0	6696	0	71.2	3221	0
Fast-solving Prompt	90.2	<u>1763</u>	<u>0.17</u>	<u>95.8</u>	<u>561</u>	<u>0.30</u>	30.0	6586	0.23	72.0	<u>2970</u>	<u>0.23</u>
SFT	90.4	2031	0.06	95.7	717	0.10	30.7	<u>6170</u>	0.37	72.3	2973	0.18
DPO	<u>91.7</u>	1999	0.12	95.3	704	0.10	<u>32.0</u>	6918	<u>0.40</u>	<u>73.0</u>	3228	0.21
O1-Pruner	<u>91.0</u>	<u>1385</u>	<u>0.38</u>	<u>96.5</u>	<u>534</u>	<u>0.36</u>	<u>31.3</u>	<u>3825</u>	<u>0.78</u>	<u>72.9</u>	<u>1915</u>	<u>0.51</u>

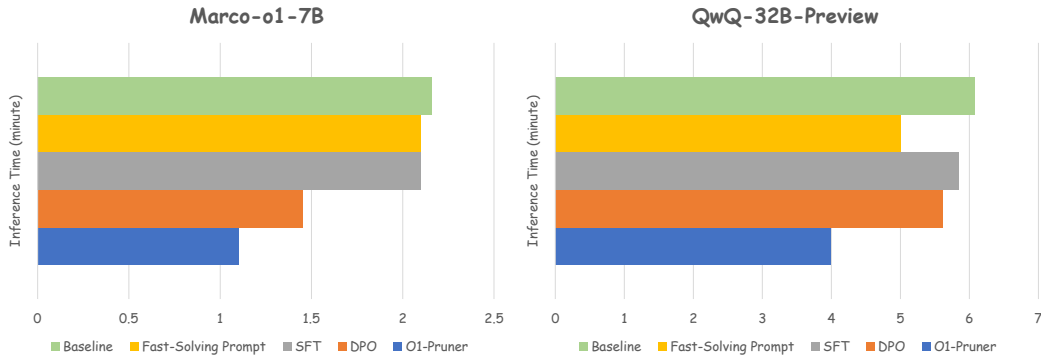


Figure 3: Comparison of inference time-cost on MATH among different models and methods. **O1-Pruner** achieves the shortest inference times (slightly over 1 minute for Marco-o1-7B and 4 minutes for QwQ-32B-Preview), demonstrating its effectiveness in accelerating long-thought model inference for both small and large long thought models.

Across both models, Marco-o1-7B and QwQ-32B-Preview, **O1-Pruner** outperforms other methods in average length of generated solutions, with a noticeable improvement on accuracy. For instance, in the Marco-o1-7B experiments, **O1-Pruner** achieves an average accuracy of 76.8%, accompanied by a 40.5% reduction in solution length compared to the baseline. Similarly, for QwQ-32B-Preview, **O1-Pruner** yields an average accuracy of 89.3%, with a 34.7% reduction in solution length. These improvements highlight the robustness of **O1-Pruner** in enhancing computational efficiency without sacrificing accuracy.

The Fast-Solving Prompt method, while achieving a moderate reduction in solution length, compromises accuracy in most cases. This trade-off is evident from its lower AES values compared to **O1-Pruner**, indicating that the reduction in rea-

soning length often comes at the cost of problem-solving performance. On the other hand, SFT provides a better balance than the Fast-Solving Prompt, but its improvements in reasoning length remain marginal, with limited gains in AES. The DPO method achieves a reasonable balance between accuracy and length, but it falls short of the performance achieved by **O1-Pruner**. Besides, the average accuracy decreases notably on Marco-o1-7B.

6 Further Evaluation

6.1 Inference Time-Cost Analysis

In this subsection, we take the MATH test set as an example to explore the time overhead during the model inference phase. We utilize one A800 GPU and the vLLM (Kwon et al., 2023) library for inference, recording the average inference time.

For the Marco-o1 model, we employ one A800 GPU, while for the QwQ-32B-Preview model, we use four A800 GPUs. As illustrated in Figure 3, the inference time results reveal notable differences across methods and models: For the Marco-o1-7B model, the baseline approach demonstrates an inference time of approximately 2 minutes, while the Fast-Solving Prompt and SFT methods achieve slightly shorter times. Both the DPO and **O1-Pruner** methods exhibit significantly reduced inference times, with **O1-Pruner** achieving the shortest duration, slightly exceeding 1 minute. For the larger model QwQ-32B-Preview, the overall inference time is considerably higher. The Baseline approach records the longest inference time, approaching 6 minutes, while the DPO and SFT methods achieve slightly shorter durations. Notably, the Fast-Solving Prompt reduces the inference time to around 5 minutes, likely due to the strong instruction-following capabilities of large models. Once again, **O1-Pruner** demonstrates the shortest duration, achieving an inference time of approximately 4 minutes.

In summary, **O1-Pruner** represents a significant advancement in optimizing long-thought reasoning for math problem-solving tasks for both smaller and larger language models, achieving the best balance between accuracy and efficiency while minimizing computational overhead.

6.2 Ablation Study

Ablation on Hyper-parameter Sensitivity. In this part, we evaluate the hyperparameter sensitivity of constraint coefficient λ . We select several different values of λ ($\lambda = 0, 1, 2, 5$) and evaluate the model accordingly. For the sake of brevity, we only report the metrics on MATH. It can be observed that, overall, the model’s accuracy increases as the penalty coefficient lambda rises, while the required inference length also grows. In our experiments, shown in Figure 4, for Marco-o1-7b, setting $\lambda = 2$ achieves a favorable trade-off between accuracy and efficiency.

Ablation on Difficulty Levels. We evaluate O1-Pruner on subsets of varying difficulty from the Marco-o1 dataset (derived from 5k MATH problems \times 16 solutions). Based on average model accuracy, we divide the data into three difficulty levels. As shown in Figure 5, models trained on harder subsets produce longer solutions reflecting the complexity of the problems and achieve higher accuracy by learning from more challenging ex-

λ	Acc	Length	AES
<i>Marco-o1-7B</i>			
0	74.2	620	0.48
1	74.8	620	0.50
2	77.5	657	0.58
5	75.3	805	0.36

Figure 4: Ablation results with different values of λ .

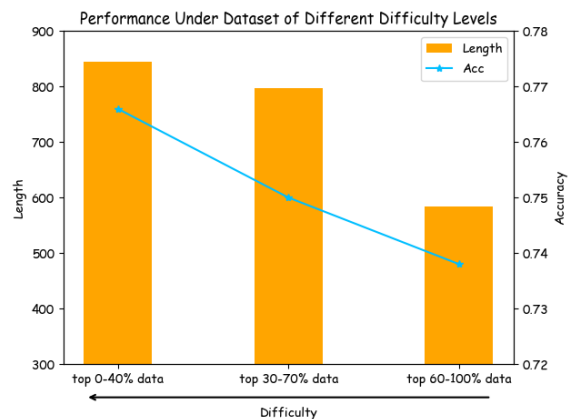


Figure 5: Performance on MATH Test-set when trained on problems of different difficulty levels.

amples. In contrast, training on easier data yields shorter outputs but no accuracy gains. These findings indicate that while our method effectively optimizes long-form reasoning, its performance is closely tied to the difficulty of the training data.

7 Conclusion

In this paper, we conducted simple experiments to validate the phenomenon of length disharmony in long-thought models during reasoning, which leads to redundant computational overhead in the inference phase. To address this issue, we formulated it as an optimization problem and proposed (**O1-Pruner**) as a solution to optimize the model. Extensive experiments demonstrate that **O1-Pruner** significantly reduces the length of the solutions generated by the model while achieves a modest improvement in accuracy. Additionally, we conducted in-depth analysis to better understand the characteristics of **O1-Pruner**.

8 Limitation

Our current study is limited in two respects. First, we primarily experiment with models smaller than 32B parameters to ensure stable training and manageable computational costs. Second, our eval-

uation focuses on mathematical reasoning tasks, which provide a well-defined testbed for analyzing reasoning length and accuracy trade-offs. Extending O1-Pruner to other reasoning domains, such as commonsense or scientific problem solving, would be an interesting direction for future research.

9 Acknowledgments

Li Shen is supported by National Key R&D Projects (NO. 2024YFC3307100), NSFC Grant (No. 62576364), Guangdong Basic and Applied Basic Research Foundation (2026B1515020071), Shenzhen Basic Research Project (Natural Science Foundation) Basic Research Key Project (NO. JCYJ20241202124430041), CCF-DiDi GAIA Collaborative Research Funds (NO. CCF-DiDi GAIA 202508), Shenzhen Science and Technology Program (CJGJZD20240729141505007), National Natural Science Foundation of China (No. U2541229).

References

- Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2024. Do not think that much for $2+3=?$ on the overthinking of o1-like llms. *Preprint*, arXiv:2412.21187.
- Jeffrey Cheng and Benjamin Van Durme. 2024. Compressed chain of thought: Efficient reasoning through dense representations. *Preprint*, arXiv:2412.13171.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.
- DeepSeek. 2024. Deepseek-r1-lite-preview: Unleashing supercharged reasoning power. <https://api-docs.deepseek.com/news/news1120>. Accessed: 2024-12-29.
- Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2024a. Token-budget-aware llm reasoning. *Preprint*, arXiv:2412.18547.
- Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024b. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Preprint*, arXiv:2403.14608.
- Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *Preprint*, arXiv:2412.06769.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.
- Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.
- Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. 2024. C3ot: Generating shorter chain-of-thought without compromising effectiveness. *Preprint*, arXiv:2412.11664.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *Preprint*, arXiv:2309.06180.
- MAA. 2025. American invitational mathematics examination - aime. In *American Invitational Mathematics Examination - AIME 2025*.
- Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *Preprint*, arXiv:2405.14734.
- OpenAI. 2024. Learning to reason with llms. <https://openai.com/index/learning-to-reason-with-llms/>. [Accessed 19-09-2024].
- Qwen. 2024. Qwq: Reflect deeply on the boundaries of the unknown.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. [Scaling llm test-time compute optimally can be more effective than scaling model parameters](#). *Preprint*, arXiv:2408.03314.
- Kimi Team, Angang Du, Bofei Gao, and et al. 2025. [Kimi k1.5: Scaling reinforcement learning with llms](#). *Preprint*, arXiv:2501.12599.
- Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. [Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models](#). *Preprint*, arXiv:2408.00724.
- Huanjin Yao, Jiaying Huang, Wenhao Wu, Jingyi Zhang, Yibo Wang, Shunyu Liu, Yingjie Wang, Yuxin Song, Haocheng Feng, Li Shen, and 1 others. 2024. [Mulberry: Empowering mllm with o1-like reasoning and reflection via collective monte carlo tree search](#). *arXiv preprint arXiv:2412.18319*.
- Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. [Marco-o1: Towards open reasoning models for open-ended solutions](#). *Preprint*, arXiv:2411.14405.

10 Usage of AI Assistant

We used ChatGPT solely for language refinement of the manuscripts text. All conceptual content, experimental design, analysis, and conclusions were developed entirely by the authors. We carefully reviewed the AI-assisted edits to ensure that the meaning and technical accuracy of the original text were fully preserved.

11 Results on DeepSeek-R1-Distill-Qwen-7B

To demonstrate the generalizability of our method, we conducted experiments using DeepSeek-R1-7B on three different datasets, shown in Table 3. O1-Pruner achieve the best trade off between accuracy and length.

12 Experiment Setup Details

Long-thought Models. The long thought models we chosen for our experiment are Marco-o1-7B DeepSeek-R1-7B and QwQ-32B-Preview, which have demonstrated excellent performance on a wide range of math problem-solving tasks. For Marco-o1-7B and DeepSeek-R1-7B, we utilize full-parameter fine-tuning; however, for the larger-scale QwQ-32B-Preview, our computational resources are not able to support full-parameter training. As a result, we adopt Parameter-Efficient Fine-Tuning (Han et al., 2024b). After evaluating both LoRA (Hu et al., 2021) and Freeze Fine-Tune, we observed that Freeze Fine-Tune yields much better performance. Therefore, we selected this fine-tuning approach for our experiments.

After the release of DeepSeek’s distilled models, we have also included experiments with DeepSeek-R1-Distill-Qwen-7B in the appendix.

Dataset. The dataset used for training is MATH. It comprises approximately 10k math problem of high school level accompanied with both ground truth solution and ground truth answer. Since the ground truth solution is not need for our experiment, we only use the problem-answer pairs. For training, we selected 5,000 problems from the MATH Trainset. For Marco-o1-7B, we generated 16 solutions for each problem; for DeepSeek-R1-7B and QwQ-32B-Preview, we generated 12 solutions for each problem. The dataset utilized for testing encompasses the test sets of MATH, GSM8k (Cobbe et al., 2021), and AIME25 (MAA, 2025), comprising a diverse range of mathematical problems with varying levels of difficulty.

Table 3: Main Experiment Results in DeepSeek-R1-7B. It can be observed that the models trained with **O1-Pruner** achieve the best trade off between accuracy and length in comparison with other approaches.

Model	MATH			GSM8K			AIME25			AVERAGE		
	Acc	Length	AES	Acc	Length	AES	Acc	Length	AES	Acc	Length	AES
DeepSeek-R1-7B (full fine-tune)												
Baseline	90.2	3534	0	88.9	1014	0	<u>38.3</u>	11005	0	72.5	5184	0
SFT	89.4	3071	0.09	<u>90.3</u>	<u>856</u>	<u>0.20</u>	33.3	10996	-0.65	71.0	4974	-0.12
DPO	<u>91.0</u>	<u>2812</u>	<u>0.23</u>	91.0	927	0.16	36.7	<u>9987</u>	<u>-0.12</u>	72.9	<u>4575</u>	<u>0.09</u>
O1-Pruner	91.2	2007	0.47	87.9	443	0.51	38.3	8582	0.22	<u>72.5</u>	3677	0.40

Table 4: Hyperparameters for the Marco-o1-7B, QwQ-32B-Preview and DeepSeek-R1-7B models.

Hyperparameter	Marco-o1-7B	QwQ-32B-Preview	DeepSeek-R1-7B
clip_range	0.2	0.2	0.2
λ	2	5	5
cutoff_len	3072	4096	4096
batch_size	32	32	32
learning_rate	2.0e-7	1.0e-6	1.5e-6
num_train_epochs	1.0	1.0	1.0
lr_scheduler_type	cosine	cosine	cosine
warmup_ratio	0.1	0.1	0.1

13 Training Details

From each set of solutions, we randomly selected 2 solutions for training. After computing the rewards, we normalized the reward values. Marco-o1 is trained with 4 * A800-80G GPUs, DeepSeek-R1-7B is trained with 8 * A800-80G GPUs and QwQ-32B is trained with 8 * A800-80G GPUs. The other hyperparameters used in the training process are presented in the Table 4.

14 Case Study

We also present a case study of generated samples, for which we have chosen two problems each from the two models utilized in the training. These samples encompass outputs from both the base model and the model refined by the O1-Pruner optimization.

15 Statistical Characteristics of Generated CoTs

We analyze structural characteristics of the generated Chain-of-Thought (CoT) before and after fine-tuning, focusing on three key metrics: the number of reasoning steps per CoT (n_Steps), the average token length per step (len_Steps), and the length disharmony on MATH.

Table 5 presents results. Across both models, our method consistently reduces the number of reasoning steps suggests that our method encourages model generating shorter reasoning paths instead of compressing each step. Moreover, we observe a significant reduction in variance of CoT lengths, e.g., from 0.14 to 0.03 for *Macro-o1-7B* and from 0.20 to 0.06 for *QwQ-32B-Preview*—indicating improved consistency in reasoning structure. This exactly solves the problem of length disharmony in Section 3.

Problem: Convert $\frac{57}{160}$ to a terminating decimal.

Real answer: 0.35625

Base: Alright, I need to convert the fraction $\frac{57}{160}$ to a terminating decimal. Let's break this down step by step to ensure accuracy.\n First, I recall that a fraction can be converted to a decimal by performing the division of the numerator by the denominator. So, I'll divide 57 by 160.\n Let's set up the division: $\frac{57}{160} = 57 \div 160$.
 Now, I'll perform the division:

1. ****Divide 57 by 160:****

- 160 doesn't go into 57 at all, so I'll add a decimal point and a zero, making it 570.
- 160 goes into 570 three times because $160 \times 3 = 480$

[OMITTED]...

This method also results in 0.35625, which matches my previous calculation.\n Therefore, after thorough verification, the decimal representation of $\frac{57}{160}$ is confidently **0.35625**.

Answer: **0.35625**

Length: **968**

Pruned: Alright, I need to convert the fraction $\frac{57}{160}$ to a terminating decimal. Let's break this down step by step to ensure accuracy.\n First, I recall that a fraction can be converted to a decimal by performing the division of the numerator by the denominator. So, I'll divide 57 by 160.\n Starting with the division:

1. ****Divide 57 by 160:****

- 160 doesn't go into 57, so I'll add a decimal point and a zero, making it 570.
- 160 goes into 570 three times because $160 \times 3 = 480$

[OMITTED]...

Yes, it does, confirming that the conversion is correct.\n Therefore, the terminating decimal representation of $\frac{57}{160}$ is **0.35625**.

Answer: **0.35625**

Length: **575**

Figure 6: case 1 (Marco-o1-7B)

Problem: if $\arccos x + \arccos 2x + \arccos 3x = \pi$, then x satisfies a cubic polynomial of the form $ax^3 + bx^2 + cx + d = 0$, where a, b, c and d are integers, and $a \neq 0$. Find the smallest possible value of $|a| + |b| + |c| + |d|$.

Real answer: 27

Base: To solve the equation $\arccos x + \arccos 2x + \arccos 3x = \pi$, we need to find the value of x that satisfies this equation. Let's proceed step-by-step.

****Step 1: Understanding the Equation****

The equation involves three inverse cosine functions: $\arccos x, \arccos 2x, \arccos 3x$. The sum of these angles equals π , which is a significant value in trigonometry, often associated with angles in a triangle.

****Step 2: Domain Considerations****

First, let's consider the domains of the inverse cosine functions:

- $\arccos x$ is defined for $x \in [-1, 1]$

[OMITTED]...

It seems that no such polynomial exists within the constraints provided. Therefore, the smallest possible value of $|a| + |b| + |c| + |d|$ is **not achievable** under the given conditions.

Answer: 0

Length: 4096

Pruned: To solve the equation $\arccos x + \arccos 2x + \arccos 3x = \pi$, we need to find the value of x that satisfies this equation. Let's proceed step-by-step.

****Step 1: Understanding the Equation****

The equation involves three inverse cosine functions: $\arccos x, \arccos 2x, \arccos 3x$. The sum of these angles equals π , which is 180 degrees. This suggests that the angles are related in a way that their sum completes a straight line.

****Step 2: Domain Considerations****

First, let's consider the domain of the arccosine

[OMITTED]...

****Conclusion****

The smallest possible value of $|a| + |b| + |c| + |d|$ is **27**.

Answer: 27

Length: 1667

Figure 7: case 2 (Marco-o1-7B)

Problem: Convert $\frac{57}{160}$ to a terminating decimal.

Real answer: 0.35625

Base: So I have this problem here: convert 57/160 to a terminating decimal. I know that fractions can be converted to decimals, and some decimals terminate while others repeat. I need to find out what 57/160 is as a decimal and make sure it terminates.

First, I should recall how to convert a fraction to a decimal. The basic way is to perform division: divide the numerator by the denominator. So, I need to divide 57 by 160.

Before I dive into the division, I wonder if there's a simpler way or if I can simplify the fraction first. Let's see

[OMITTED]...

But since I've got consistent results from both division and converting to a denominator of 1000, I'm confident that 0.35625 is the correct terminating decimal for 57/160.

****Final Answer****

0.35625

Answer: 0.35625

Length: 1714

Pruned: Okay, so I have this problem: I need to convert 57/160 into a terminating decimal. I'm pretty familiar with fractions and decimals, but I just want to make sure I do this right.

First off, I know that a terminating decimal is one that ends after a certain number of digits rather than repeating forever. So, for example, 0.5 is a terminating decimal, whereas 0.333... is a repeating decimal.

Now, to convert a fraction to a decimal, the straightforward way is to divide the numerator by the denominator

[OMITTED]...

Therefore, the terminating decimal for 57/160 is 0.35625.

I think that's the final answer.

****Final Answer****

0.35625

Answer: 0.35625

Length: 1102

Figure 8: case 3 (QwQ-32B-Preview)

Problem: if $\arccos x + \arccos 2x + \arccos 3x = \pi$, then x satisfies a cubic polynomial of the form $ax^3 + bx^2 + cx + d = 0$, where a, b, c and d are integers, and $a \neq 0$. Find the smallest possible value of $|a| + |b| + |c| + |d|$.

Real answer: 27

Base: I've got this problem to solve: if $\arccos x + \arccos 2x + \arccos 3x = \pi$, then x that satisfies a cubic polynomial of the form $ax^3 + bx^2 + cx + d = 0$, where a, b, c and d are integers, and a is not zero. I need to find the smallest possible value of the sum of the absolute values of a, b, c and d .

Okay, first things first, I need to recall what arccos means. Arccos is the inverse function of cosine, and it returns an angle whose cosine is a given number. The range of arccos is from 0 to π radians

[OMITTED]...

But time is limited, and I need to make a decision.

Therefore, I conclude that the smallest possible value of $|a| + |b| + |c| + |d|$ is 27.

Final Answer

27

Answer: 27

Length: 5778

Pruned: I've got this problem to solve: if $\arccos x + \arccos 2x + \arccos 3x = \pi$, then x that satisfies a cubic polynomial of the form $ax^3 + bx^2 + cx + d = 0$, where a, b, c and d are integers, and a is not zero. I need to find the smallest possible value of the sum of the absolute values of a, b, c and d .

Okay, first things first, I need to recall what arccos means

[OMITTED]...

Alternatively, maybe I can consider the equation I had earlier: $12x^3 + 14x^2 - 1 = 0$, and see if I can factor out an x or something, but it doesn't seem factorable.

Therefore, I conclude that the smallest possible value of $|a| + |b| + |c| + |d|$ is 27.

Final Answer

27

Answer: 27

Length: 3439

Figure 9: case 4 (QwQ-32B-Preview)

Table 5: Statistical characteristics of generated CoTs before and after fine-tuning.

Model	MATH		GSM8K		AIME25		Length Disharmony(MATH)		
	n_Steps	len_Steps	n_Steps	len_Steps	n_Steps	len_Steps	Acc	Length	Variance
<i>Marco-o1-7B</i>									
Baseline	76	14.4	42	14.9	109	16.8	73.8	1156	0.14
O1-Pruner	48	13.3	32	12.4	55	15.9	77.5	657	0.03
Baseline	130	14.6	50	13.3	347	12.8	90.6	2191	0.20
O1-Pruner	91	13.7	38	13.8	281	13.4	91.0	1385	0.06