

# Step Guided Reasoning: Improving Mathematical Reasoning using Guidance Generation and Step Reasoning

Anonymous ACL submission

## Abstract

Mathematical reasoning has been challenging for large language models (LLMs). However, the introduction of step-by-step Chain-of-Thought (CoT) inference has significantly advanced the mathematical capabilities of LLMs. Despite this progress, current approaches either necessitate extensive inference datasets for training or depend on few-shot methods that frequently compromise computational accuracy. To address these bottlenecks in mathematical reasoning, we propose a novel method called Step Guided Reasoning, which is more stable and generalizable than few-shot methods and does not involve further fine-tuning of the model. In this approach, LLMs reflect on small reasoning steps, similar to how humans deliberate and focus attention on what to do next. By incorporating this reflective process into the inference stage, LLMs can effectively guide their reasoning from one step to the next. Through extensive experiments, we demonstrate the significant effect of Step Guided Reasoning in augmenting mathematical performance in state-of-the-art language models. Qwen2-72B-Instruct outperforms its math-specific counterpart, Qwen2.5-72B-Math-Instruct, on MMLU-STEM with a score of 90.9%, compared to 87.3%. The average scores of Qwen2-7B-Instruct and Qwen2-72B-Instruct increase from 27.1% to 36.3% and from 36.5% to 47.4% on the mathematics domain, respectively.

## 1 Introduction

Since the introduction of Chain-of-Thought (CoT) (Wei et al., 2022) reasoning on LLMs (Yang et al., 2024c; Zhao et al., 2023; Vaswani et al., 2017), it has been demonstrated how reasoning abilities naturally emerge in sufficiently large language models through a simple technique called thought chaining prompts. This approach involves enriching the prompts (Sahoo et al., 2024) with thought chaining examples, which serve as demonstrations to guide the model’s reasoning process. However,

complex mathematical reasoning remains a significant challenge for LLMs (He et al., 2024a). Even though the accuracy of LLMs in mathematical reasoning can be improved with the scaling of model parameters and that of the training data, the amount of high-quality CoT data (Cheng et al., 2024) becomes the bottleneck (Hoffmann et al., 2022).

There are several approaches to tackle these challenges in the inference stage, and the methods discussed below significantly enhance the model’s performance on both mathematical reasoning and MMLU-STEM benchmarks (Hendrycks et al., 2021a). Cumulative reasoning (Zhang et al., 2023) has been proposed to make great improvements over MATH datasets (Hendrycks et al., 2021b). Cumulative reasoning significantly enhances problem-solving by decomposing the task into smaller, more manageable elements and builds upon prior propositions, improving the overall effectiveness of problem-solving. Additionally, Zheng et al. proposed a “Take a Step Back” method, which introduced overall concepts and principles to guide model reasoning using results from high-level descriptions of original questions. Both of these schemes improve the accuracy of mathematical reasoning by generating intermediate but useful contexts, namely “scratchpad” (Nye et al., 2021), during the inference phase.

Another approach to enhancing mathematical reasoning ability involves methods that increase computation during the inference stage (Zhang et al., 2024; Gao et al., 2024; Yao et al., 2024; Snell et al., 2024). These approaches enable LLMs to explore multiple possible reasoning paths and select the most likely correct ones. To be more specific, techniques such as Best-of-N (BoN) (Cobbe et al., 2021; Dong et al., 2023) and Tree-of-Thought (ToT) (Yao et al., 2024) have also been explored. By scoring intermediate reasoning steps or evaluating the entire final result, the highest-scoring outcome by the reward model (RM) (Ouyang et al.,

2022) is selected as the final answer. These strategies have been shown to effectively improve the model’s mathematical reasoning ability, allowing it to tackle more complex problems with better accuracy and reliability.

However, when coming to challenging math problems, such as competition-level math (AI-MO, 2024b,a), neither of them works well, and approaches like ToT and BoN require an additional reward model for scoring. To be more specific, we observed that more challenging math tasks often require more thoughtful reasoning steps (see analysis of Figure 2) to complete the answer. Inspired by these observations, we propose a method called Step Guided Reasoning (SGR) that introduces guided thought in step-by-step reasoning, and SGR can improve challenging math problems without finetuning (Parthasarathy et al., 2024) the model and without the need for a reward model like BoN. In each reasoning step of our approach, the model is prompted to self-question (Renze and Guven, 2024) what to do next, self-answer this question which can help the next-step generation, and use this reflection to guide the subsequent generation process. Through this method, we observed substantial improvements in solving complex problems, particularly in multi-step tasks such as the Olympic Mathematical Challenge (He et al., 2024b), where the model already demonstrates a certain level of accuracy on the test dataset through 0-shot CoT (Kojima et al., 2022).

By applying our method, Qwen2-7B-Instruct improved the accuracy on the MATH dataset Level 5 (Hendrycks et al., 2021b), the most difficult level, from 37.1% to 58.6%, while Qwen2-Math-7B-Instruct achieved an accuracy of 52.0%. Similarly, Qwen2-72B-Instruct achieved an improvement from 35.8% to 41.2% on the OlympiaBench (He et al., 2024a) open-ended, no-image English Math Competition test set, with Qwen2-Math-72B-Instruct achieving an accuracy of 42.5%.

To summarise our contributions, we present a generalized approach that facilitates autonomous inference strategies in mathematical reasoning without requiring fine-tuning in the domain of mathematical logic, leading to substantial improvements on challenging math datasets.

## 2 Method

Step Guided Reasoning (SGR) method employs a series of reasoning steps during inference, each

step consisting of generating two key components: a *step guidance* and a *step answer*.<sup>1</sup> The *step guidance* distills the most crucial logical elements and generates inferential cues. Functioning as a more sophisticated prompt signal, it fortifies every reasoning step. The *step answer* then harnesses these cues comprehensively to yield more refined intermediate step responses. As a result, the overall reasoning becomes more efficient and impactful.

As illustrated in Figure 1, SGR incorporates a multi-round iterative reasoning mechanism. At the first iteration (Stage-I) of the reasoning, upon receiving a math query, we first direct the model to formulate a *Step Guidance Question*. Subsequently, we prompt the model to engage in in-depth deliberation and response, thus eliciting a *step guidance*. This enables the model to generate a high-quality *step answer* autonomously. In the following iterative cycles (Stage-II), we gradually leverage the step answer obtained from the preceding round to refine the step answer at the  $k$ -th step, until the model outputs a satisfactory result.

SGR method provides a simple guidance mechanism effectively promotes the model’s thinking process and significantly enhances its reasoning capabilities. By following this multi-round iterative reasoning mechanism, the model can break down complex mathematical problems into more manageable steps, leading to more accurate and logical reasoning.

### 2.1 Reasoning Step

SGR consists of multiple iterations as the *reasoning steps* to instruct LLMs during inference. As shown in Figure 1, the first step initiates a reasoning cycle (Stage I), and the subsequent steps (Stage II) iteratively refine the current step answer. Each "step" can be defined at various granularities, including token-level (Zelikman et al., 2024), sentence-level (Jarrahi et al., 2023), paragraph-level (Chalkidis et al., 2021; Zhang et al., 2021), or block-level, typically annotated by human experts (Lightman et al., 2024). In this paper, we opt to define a step as a paragraph level, since our approach focuses on challenging mathematical problems which generally require answers spanning thousands of tokens (Fu et al., 2023). Selecting appropriate granularity for math domain ensures the effectiveness of instructing without losing coherence or logical flow while minimizing computa-

<sup>1</sup>All used prompts are listed in Appendix A.1.

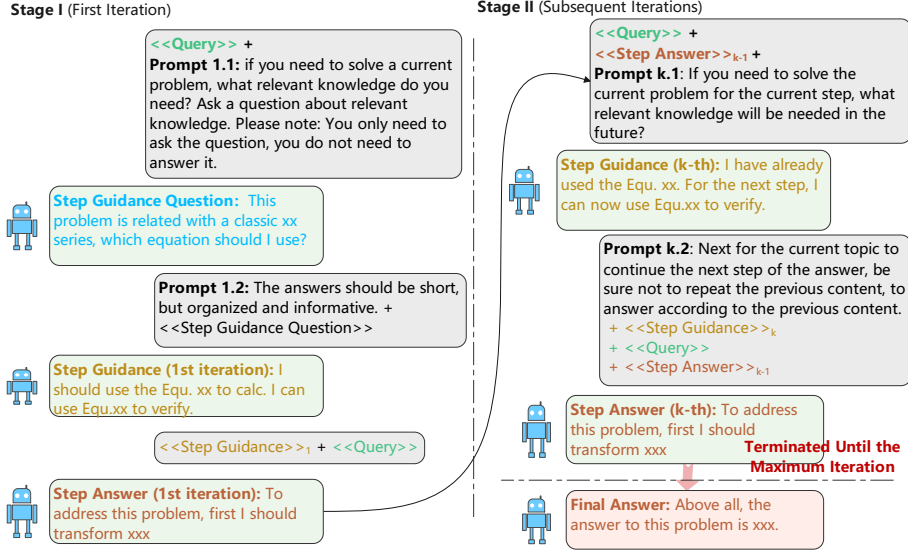


Figure 1: Illustration of how our proposed SGR method generates **step guidance** and **step answer** for each iteration  $k$ . In stage I ( $k = 1$ ), **Prompt 1.1** questions the model to search for relevant knowledge. Subsequently, **Prompt 1.2** elicits a guidance from the model by getting it to answer the **step guidance question**. Original **query** with such a **step guidance** empowers the model to generate a more accurate and well-reasoned **step answer**. In stage II ( $1 < k \leq N$ ), the **step answer** at step  $k$  is refined by reiterating the process from **step answer**  $k - 1$  with **Prompt k.1** and **k.2**. We iteratively enhance the **step answer** until a satisfactory final answer is obtained.

tional overhead.

In practice, we found delimiter “. \n\n” serves as an effective boundary for logical inference for most instruct models, such as GPT-4/GPT-4o, Qwen, and LLaMA. However, directly splitting reasoning at every occurrence of “. \n\n” can lead to repeated patterns in the model generation, causing the model to reanalyze the first step instead of progressing to the next. This issue arises because the model may interpret each split as a signal to re-analyze the problem, rather than advancing through the reasoning process.

To mitigate this problem, we introduce a step length constraint, where each step, delimited by “. \n\n”, must contain a minimum number of characters. This helps ensure that each step contains sufficient information for meaningful reasoning and reduces the tendency for the model to repeat earlier analyses. Although this constraint addresses some of the repetition, LLMs could still exhibit long repetitive patterns in subsequent steps by chance, which would be fixed by fine-tuning to improve instruction following.

In theory, the step length required for different problems may vary, and even within a single problem, the length of steps may differ depending on the complexity of the reasoning required. Ideally, fine-tuning the language models over manually labelled data with a special step token could explicitly dis-

tinguish between steps, providing further clarity and precision in the reasoning process. However, this approach is not considered in the current paper, as our focus remains on leveraging an instruction-based model that requires no additional fine-tuning.

## 2.2 Step Guidance

For each iteration, the prompt guides the LLM to think about what relevant knowledge is needed next as *step guidance*, and the model is then asked to generate the corresponding reasoning as the *step answer*. The model does not revisit or retain *previous step guidance*; instead, each generated *step guidance* is used exclusively for the current step, ensuring that each step is handled independently without carrying over unnecessary context.

For the first iteration, we adopt the SBP approach (Zheng et al., 2024) by using a question to obtain a more general *step guidance*. Specifically, in the first iteration, the model is prompted to independently generate a question related to the query as the *step guidance question*, and then the LLM answers this *step guidance question*, with the answer serving as the *step guidance*.

## 2.3 Step Answer

To generate the result of  $k$ -th reasoning step, both the generated *step guidance* at step  $k$  and the previously accumulated  $\langle\langle \text{step answer} \rangle\rangle_{k-1}$  are

incorporated into the prompt to support continued reasoning. The generation process is halted once the model reaches the token “. \n\n” with a minimum length, which indicates the completion of the current step. This serves as a natural delimiter, ensuring that each step is sufficiently detailed and self-contained.

To ensure the quality of generation of the  $\ll$  step answer  $\gg_k$ , we explicitly emphasized that "not to repeat the previous content" in the **Prompt k.2**. However, such repetitions still occurred. To address this, whenever a duplicate of the current step is detected, it is removed and the model is prompted to resample and generate a new response. This trick ensures a streamlined reasoning process that eliminates unnecessary repetition, enabling the model to advance smoothly through each step without redundancy.

Unlike Retrieval-Augmented Generation (RAG) (Gao et al., 2023), which leverages additional pre-existing or externally-retrieved context to enhance reasoning, our step answer mechanism hinges on step guidance where the additional context is generated autonomously by the Large Language Model (LLM) itself, rather than being sourced from external repositories. This unique characteristic of dynamic context generation endows our method with greater flexibility and adaptability during real-time reasoning processes, enabling the model to adjust its reasoning strategies according to the evolving requirements of the task at hand.

### 3 Experiments

#### 3.1 Experimental Setups

**Datasets** For evaluation, we use four representative challenging math benchmarks, AMC23 (AI-MO, 2024a), MATH (Hendrycks et al., 2021b), AIME24 (AI-MO, 2024b) and OlympiadBench (OLY) (He et al., 2024b) with the open-ended, no-image English Math Competition (OE\_TO\_maths\_en\_COMP) tag. The selected mathematics test sets are all challenging and include competition-level questions (See A.2).

To assess the scalability of our method—whether it can also be effective in domains beyond mathematical logical reasoning—we selected MMLU (Hendrycks et al., 2021a) with STEM tags (MMLU-STEM) for evaluation. STEM, which encompasses the fields of Science, Technology, Engineering, and Mathematics, often requires spe-

cialized problem-solving skills. Each of the four datasets provides the problem as a query along with a reference answer, and we report the accuracy by comparing the final output of the LLM with the reference answer. Specifically, for the MMLU-STEM test dataset, a multiple-choice dataset, we determine accuracy by comparing the final selected answer option with the reference answer. For the other test sets, we first accurately extract the final answer from the reference answer and then compare this extracted final answer with the answer generated by the model to ensure that the model’s output aligns with the intended task objectives. To ensure the reliability and consistency of our evaluation, we employ GPT-4 (OpenAI et al., 2024) as our validation tool, a model that has demonstrated near-human-level evaluation capabilities (Sottana et al., 2023).

**Models** Given that the SGR method demands that LLMs display remarkably strong and comprehensive capabilities, we choose Qwen2-72B-Instruct, Qwen2-7B-Instruct (Yang et al., 2024a), LLaMA3.1-8B-Instruct (Dubey et al., 2024) and LLaMA2-70B-Instruct (Touvron et al., 2023) as our experimental model. We also use a distilled version of DeepSeek-R1 of Qwen-7b and LLaMA2-8b (DeepSeek-AI et al., 2025) to compare with Qwen-7b and LLaMA2-8B as the base instruct models promoted by our method.

We then compared our method to the state-of-the-art models QwQ-32B-Preview (Team, 2024), Qwen2-Math-7B-Instruct, Qwen2-Math-72B-Instruct (Yang et al., 2024a), Qwen2.5-Math-7B-Instruct, Qwen2.5-Math-72B-Instruct (Yang et al., 2024b), and GPT-4o (OpenAI, 2023). Note that we do not use these models because they are specifically fine-tuned for mathematics, which has led to a loss of its instruction-following capabilities. Therefore, this disables the model to follow guidance effectively.

Alongside the 0-shot CoT results for LLMs, we also provide a comparison with two representative methods: Best-of-N (BoN) (Cobbe et al., 2021) and “Take a Step Back Prompt”(SBP) (Zheng et al., 2024). For the BoN method, sampling 16 or 32 times for each problem using Qwen2-7B-Instruct. The Qwen2.5-Math-RM-72B (Yang et al., 2024b) model was then used to score these responses, with the one receiving the highest score selected as the final result. For SBP, we utilized the original prompt template from the SBP method along with the ex-

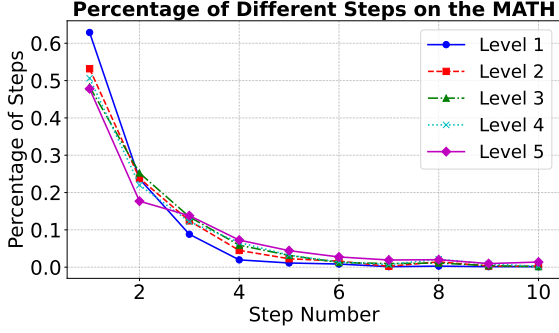


Figure 2: This figure illustrates the proportion of different steps at which the correct answer first appears for problems across various difficulty levels (Level) in the MATH dataset. The result represents the average accuracy of the outputs from Qwen2-7b-Instruct with top\_p values of 0.7 and 1.0.

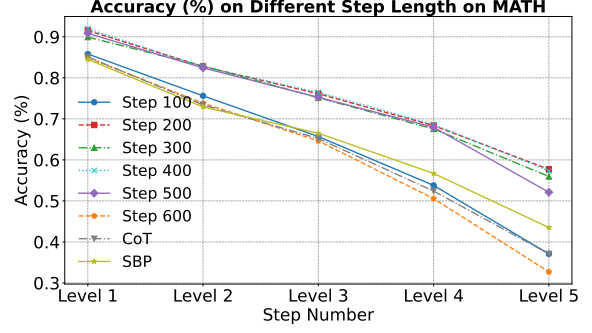


Figure 3: As the change of step length thresholds, our SGR accuracy on the MATH dataset by Qwen2-7B-Instruct. The 0-shot Chain of Thought (CoT) and Step-Back Prompt (SBP) generated by the same model are compared as the baseline.

ample provided in the appendix to construct a 5-shot prompt. This prompt was employed to generate both the principal and the final answer.

**Hyperparameters** For the decoding strategy, we use temperature as 1.0 and set top\_p to 1.0 and 0.7 for sampling.<sup>2</sup> All experimental results are reported as the average accuracy scores under top\_p values of 0.7 and 1.0. The step length constraint for MATH and MMLU-STEM was specified as 300, while for the AIME24 dataset, it was set to 500. We use a maximum of 10 iterations for all test sets. If there is a duplication between steps, it will delete and re-sample the solution in the current step. We conducted the experiment using 8 V100 GPUs, with each problem in the test dataset generating an average output of 6,384 tokens from the MATH dataset by the Qwen2-7B-Instruct. We use float32 precision for the LLaMA3.1-8B-Instruct/Qwen2-7B-Instruct model, but float16 precision for the Qwen2-72B-Instruct model, leading to some degree of performance degradation. The native float16 precision is utilized for the LLaMA2-70B-Instruct model.

### 3.2 Experimental Results

Table 1 shows comparison results of our method and the SOTA, demonstrating distinct performance of SGR across different datasets. In the MATH dataset, SGR improves more than 10% over 0-shot CoT, with the exception of LLaMA2-70B-Instruct model. On hard reasoning tasks, (like MATH L5), SGR greatly stimulates the potential of LLaMA3.1-

8B-Instruct, leading to 188% improvement. Unlike BoN, which exhibits uniform growth across the entire difficulty spectrum of the test set across different domains, SGR outperforms methods like BoN on more difficult test sets (MATH Level 4 and Level 5). It shows that our method effectively enhanced the mathematical reasoning on hard problems. Although these base models show poor performances on AIME24 dataset, applying SGR again lifts the reasoning abilities.

Beyond the math domain, we also test our method on the MMLU-STEM dataset, showing that SGR method consistently achieves the SOTA. In Table 1, SGR outforms than BoN methods consistently on Qwen2-7B-inst model. The accuracy of Qwen2-72B-Instruct + our method reaches 90.9%, which is comparable to QwQ-32B-Preview (91.8%), a strong baseline fine-tuned with slow thinking. Moreover, we conduct a further investigation into the slow-thinking-enhanced Qwen2-7B and LLaMA3.1-8B models that incorporate distilled data from DeepSeek-R1. As demonstrated in Table 3, our method confers greater capabilities to the vanilla instruction models compared to the distilled models, which acquire advanced reasoning abilities from a powerful pre-trained model, highlighting the superiority and effectiveness of our proposed method.

### 3.3 Analysis

**Step Analysis** We analyse the number of steps to the first appearance of the answer in different levels (see Figure 2) on MATH. As illustrated in the chart, higher-level problems generally require more steps to reach a final solution compared to lower-level problems.

<sup>2</sup>We observed that top\_p decoding tends to mitigate repetition compared with greedy decoding.

Method		MATH						AMC23	AIME24	OLY	Average
		L1	L2	L3	L4	L5	Average				
Slow-Thinking Model											
QwQ-32B-Preview		97.5	96.4	95.4	91.8	84.9	92.2	85.0	50.0	67.4	73.7
Math-Specific Models											
Qwen2-Math-7b-inst		93.1	87.2	82.6	72.4	52.0	73.8	62.5	13.3	34.1	45.9
Qwen2-Math-72b-inst		95.0	94.1	90.5	83.7	67.7	83.9	60.0	20.0	42.5	51.7
Qwen2.5-Math-7b-inst		95.4	93.0	89.7	82.7	67.4	83.2	62.5	33.3	37.3	54.1
Qwen2.5-Math-72b-inst		96.3	93.5	90.9	84.9	73.3	85.7	70.0	43.3	60.6	65.5
General Models											
GPT-4o	CoT	95.0	91.7	86.0	74.9	53.8	76.6	15.0	10.0	43.3	36.2
	SBP	91.3	88.3	81.1	71.5	51.2	73.0	15.0	6.7	43.3	34.4 (-1.8)
Qwen2-7b-inst	CoT	85.1	73.4	65.2	52.4	37.1	57.8	28.8	1.5	20.1	27.1
	SBP	84.2	71.8	64.1	52.1	38.4	57.5	22.5	0.0	27.3	26.8 (-0.3)
	SGR	90.2	81.3	74.6	68.3	58.6	71.4	38.8	1.5	33.3	36.3 (+9.2)
	BoN@16	91.5	84.6	76.4	62.7	40.3	66.4	46.3	5.0	31.7	37.4 (+10.3)
	BoN@32	92.8	85.5	79.7	66.8	44.5	69.4	52.5	10.0	34.4	41.6 (+14.5)
Qwen2-72b-inst	CoT	91.4	85.3	77.3	66.9	46.1	69.2	35.0	6.0	35.8	36.5
	SBP	88.6	82.2	72.1	60.2	38.7	63.6	36.3	1.7	32.7	33.6 (-2.9)
	SGR	93.9	89.3	83.7	76.9	65.6	79.2	61.3	8.0	41.2	47.4 (+10.9)
LLaMA3.1-8b-inst	CoT	76.2	61.2	50.8	36.6	21.2	43.7	20.0	8.0	14.4	21.5
	SBP	75.3	59.3	48.1	36.4	21.2	42.5	11.3	5.0	18.5	19.3 (-2.2)
	SGR	81.7	76.8	71.5	66.8	61.2	69.5	18.8	6.0	22.7	29.2 (+7.7)
LLaMA2-70b-inst	CoT	44.5	25.4	15.8	9.6	5.2	15.7	4.0	0.0	2.3	5.5
	SBP	39.8	26.1	19.1	14.8	14.7	19.9	6.3	0.0	5.1	7.8 (+2.3)
	SGR	38.7	25.3	16.8	11.3	7.1	16.3	5.0	3.3	2.7	6.8 (+1.3)

Method		MMLU-STEM						
		Physics	Chemistry	Biology	Computer Science	Math	Engineer	Average
Slow-Thinking Model								
QwQ-32B-Preview		93.9	83.1	94.0	88.8	95.1	86.1	91.8
Math-Specific Models								
Qwen2-Math-7b-inst		69.1	57.5	64.4	65.3	84.3	62.5	71.5
Qwen2-Math-72b-inst		87.3	78.1	88.1	81.9	90.7	79.9	86.2
Qwen2.5-Math-7b-inst		71.3	61.1	61.9	66.7	86.8	61.1	73.0
Qwen2.5-Math-72b-inst		88.2	78.7	86.9	83.9	92.6	81.2	87.3
General Models								
GPT-4o	CoT	90.0	64.8	94.7	85.3	87.8	83.3	86.1
	SBP	89.6	82.1	95.1	87.0	87.9	77.8	87.8 (+1.7)
Qwen2-7b-inst	CoT	65.9	56.0	79.5	64.7	73.2	62.2	64.9
	SBP	65.4	54.7	76.2	65.2	70.6	65.3	67.5 (+2.9)
	SGR	79.2	72.3	88.9	85.2	84.1	74.0	82.3 (+17.4)
	BoN@16	67.9	56.1	80.0	66.1	82.1	59.7	73.0 (+8.1)
	BoN@32	71.2	60.8	82.2	67.2	83.9	61.5	75.4 (+10.5)
Qwen2-72b-inst	CoT	86.3	74.9	93.8	81.8	86.5	75.3	85.3
	SBP	81.8	70.6	91.4	80.3	82.7	71.9	81.5 (-3.8)
	SGR	90.7	83.2	95.1	91.3	92.7	78.8	90.9 (+5.6)
LLaMA3.1-8b-inst	CoT	59.4	62.4	56.1	78.4	61.2	64.9	69.2
	SBP	62.7	57.7	77.6	60.2	65.4	65.7	64.9 (-4.3)
	SGR	77.7	82.1	78.6	89.2	85.9	81.1	82.4 (+13.2)
LLaMA2-70b-inst	CoT	46.0	39.4	72.0	55.9	38.7	51.8	48.1
	SBP	63.4	58.7	75.3	63.2	52.1	61.5	60.3 (+12.2)
	SGR	69.3	62.3	83.1	75.3	57.9	71.5	67.3 (+19.2)

Table 1: Accuracy comparison (%) of CoT, SBP(5-shot) and our SGR methods with the SOTA over MATH (Level 1 to Level 5), AMC23, AIME24, MMLU-STEM and OLY datasets. We also report the results of open-sourced SOTA math-specific models - the QwQ, Qwen-Math models and GPT-4o. The best results are in **Bold** for each base and **Red** denotes the highest score in the current test set. **Green** indicates lower results compared to CoT, while **Red** denotes higher results.

	Method	MATH						OLY	AMC23	AIME24	Average
		L1	L2	L3	L4	L5	Average				
Qwen2-7b-inst	CoT	85.1	73.4	65.2	52.4	37.1	57.8	20.1	28.8	1.5	27.1
	Stage I	84.5	72.9	66.5	56.7	43.5	60.8	20.7	32.5	3.0	29.3 (+2.2)
	Stage II	88.6	77.7	68.8	58.4	40.1	62.3	40.9	27.5	0	32.7 (+5.6)
	SGR	90.2	81.3	74.6	68.3	58.6	71.4	33.3	38.8	1.5	36.3 (+9.2)
Qwen2-72b-inst	CoT	91.4	85.3	77.3	66.9	46.1	69.2	35.8	35.0	6.0	36.5
	Stage I	88.1	80.4	74.1	62.7	46.8	66.5	31.6	37.5	5.0	35.2 (-1.3)
	Stage II	93.9	87.6	82.1	71.6	53.9	74.0	50.0	45.0	6.7	43.9 (+7.4)
	SGR	93.9	89.3	83.7	76.9	65.6	79.2	41.2	61.3	8.0	47.4 (+10.9)
LLaMA3.1-8b-inst	CoT	76.2	61.2	50.8	36.6	21.2	43.7	14.4	20.0	8.0	21.5
	Stage I	69.1	53.6	45.3	33.5	22.6	40.1	12.6	18.8	8.0	19.9 (-1.6)
	Stage II	77.6	66.0	55.6	43.5	27.6	49.1	26.8	23.8	5.0	26.2 (+5.1)
	SGR	81.7	76.8	71.5	66.8	61.2	69.5	22.7	18.8	6.0	29.3 (+7.8)
LLaMA2-70b-inst	CoT	44.5	25.4	15.8	9.6	5.2	15.7	2.3	4.0	0.0	5.5
	Stage I	34.8	18.6	11.2	6.0	3.1	11.2	3.8	0.0	2.7	4.4 (-1.1)
	Stage II	43.6	27.9	17.4	12.1	6.4	17.4	7.5	8.3	4.1	9.3 (+3.8)
	SGR	38.7	25.3	16.8	11.3	7.1	16.3	2.7	5.0	3.3	6.8 (+1.3)

	Method	MMLU-STEM						
		Physics	Chemistry	Biology	Computer Science	Math	Engineer	Average
Qwen2-7b-inst	CoT	65.9	56.0	79.5	64.7	73.2	62.2	64.9
	Stage I	65.7	55.1	77.7	65.0	72.5	58.3	62.9 (-2.0)
	Stage II	77.0	71.6	85.6	84.2	84.9	73.6	81.0 (16.1)
	SGR	79.2	72.3	88.9	85.2	84.1	74.0	82.3 (17.4)
Qwen2-72b-inst	CoT	86.3	74.9	93.8	81.8	86.5	75.3	85.3
	Stage I	84.8	71.1	90.7	79.8	83.6	70.5	82.9 (-2.4)
	Stage II	92.6	88.4	95.6	92.7	92.0	79.9	91.5 (+6.2)
	SGR	90.7	83.2	95.1	91.3	92.7	78.8	90.9 (+5.6)
LLaMA3.1-8b-inst	CoT	59.4	62.4	56.1	78.4	61.2	64.9	69.2
	Stage I	59.7	61.4	54.0	77.0	62.0	60.9	67.9 (-1.3)
	Stage II	82.8	77.3	91.7	87.8	82.4	79.9	83.7 (+14.5)
	SGR	77.7	82.1	78.6	89.2	85.9	81.1	82.4 (+13.2)
LLaMA2-70b-inst	CoT	46.0	39.4	72.0	55.9	38.7	51.8	48.1
	Stage I	49.9	40.0	74.2	55.7	37.1	55.6	48.9 (+0.9)
	Stage II	71.3	65.0	85.1	76.5	61.7	75.4	70.0 (+21.9)
	SGR	69.3	62.3	83.1	75.3	57.9	71.5	67.3 (+19.2)

Table 2: Accuracy(%) results for Qwen2-7B-Instruct, Qwen2-72B-Instruct, LLaMA3.1-8B-Instruct and LLaMA2-70B-Instruct using different prompting methods on MATH, AMC23, AIME24, OLY and MMLU-STEM test datasets. The stage I refers to the initial iteration within SGR framework (0-shot). The stage II is the second SGR involves enhancing the first iteration by prompting the model from the outset to decide what action to take next. For this part of the experiment, we utilized a top\_p sampling method with a value of 0.7 and 1.0. We report the average of the accuracy. The best results are in **Bold** for each base and **Red** denotes the highest score in the current test set. **Green** indicates lower results compared to CoT, while **Red** denotes higher results.

**Comparison of Token Numbers** Figure 4 illustrates the relationship between the average number of tokens per query and the accuracy generated by the Qwen2-7B-Instruct model on the MATH and MMLU-STEM test sets using different methods. It is evident that with our method, we achieve better results than BoN@32 while using less than half the number of tokens on MATH.

**Step Length Thresholds Analysis** We evaluate model performance using different step lengths, ranging from 100 to 600, on the MATH dataset.

As illustrated in Figure 3, we observe that for step lengths ranging from 200 to 400, the accuracy is significantly higher compared to the baseline, with only minor variations in accuracy across this range. The step length serves as a crucial hyperparameter, where the exact split point is dynamically determined by the first occurrence of the sequence “. \n\n” following the initially specified step length. When the step length varies between 200 and 400, the results of step division show little difference.

	Method	MMLU-STEM						
		Physics	Chemistry	Biology	Computer Science	Math	Engineer	Average
Qwen2-7b-inst	CoT	65.9	56.0	79.5	64.7	73.2	62.2	64.9
	SGR	79.2	72.3	88.9	85.2	84.1	74.0	82.3 (+17.4)
DeepSeek-R1-Distill-Qwen-7b	CoT	81.0	75.1	71.7	72.4	90.8	72.2	80.6 (+12.3)
LLaMA3.1-8b-inst	CoT	59.4	62.4	56.1	78.4	61.2	64.9	69.2
	SGR	77.7	82.1	78.6	89.2	85.9	81.1	82.4 (+13.2)
DeepSeek-R1-Distill-Llama-8b	CoT	74.9	75.1	81.2	70.7	82.5	65.3	77.2 (+12.3)

Table 3: This figure compares the MMLU-STEM accuracy (%) of LLaMA3.1-8B-series and Qwen2-7B-series under three conditions: (1) the Chain of Thought (CoT) results using the instruct model as baseline, (2) the results after applying the SGR method through instruct models, and (3) the performance following distillation with DeepSeek-r1 (DeepSeek-AI et al., 2025). The best results are in **Bold** for each base and **Red** denotes the highest score in the current test set. **Green** indicates lower results compared to CoT, while **Red** denotes higher results.

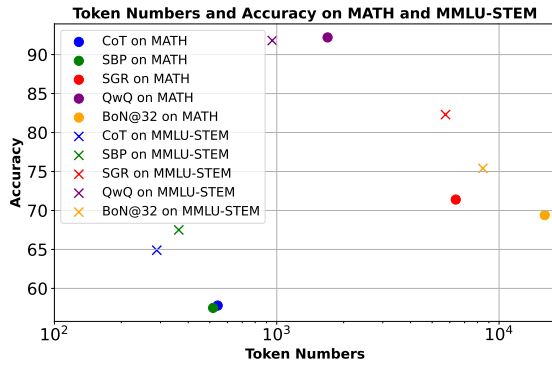


Figure 4: The scatter plot shows the relationship between the token numbers per query and accuracy for the MATH and MMLU-STEM datasets by Qwen2-7B-Instruct in different methods and QwQ-32B-Preview.

**Case Study** To understand how our method improves the reasoning procedure, we demonstrate an example in Figure 5. Compared to CoT at step 1, when calculating the "second train", the step guidance generated by SGR can help the model to carry out the correct logical reasoning, while CoT reasoning makes an error. The full contents of this example are included in the Appendix A.3.

### 3.4 Ablation

As shown in Figure 1, to explore the impact of each individual component, we evaluate the results of using each stage independently. Therefore, our approach is divided into two stages. In stage I, we prompt the LLMs to ask a step guidance question without employing a few-shot template, allowing the model to answer the step guidance question directly as the step guidance. In stage II, we directly ask the model what knowledge it needs to use next and continue the process iteratively as step

guidance. All results are presented in Table 2.

When we check the step guidance and step answer, we find that for particularly challenging problems (OLY), the LLM struggles to generate the step guidance question, often repeating the query. This severely undermines the effectiveness of step guidance. However, when the LLM is allowed to directly use the prompt from Stage II to generate step guidance, the quality of the step guidance is significantly improved compared to Stage I. As a result, the OLY achieved higher accuracy using only Stage II, outperforming the full SGR. However, we do not consider Stage I to be ineffective. This is because, compared with using the complete SGR method, it can bring about a more significant improvement in the overall performance in MATH.

## 4 Conclusion

We propose a step-by-step reasoning method that incorporates guidance generation within each step for multiple problem tasks. Our method, applicable to general instruction LLMs without the need for further fine-tuning, employs self-questioning and self-answering at each reasoning step, where the model generates and answers to guide the step answer, enhancing the overall reasoning process. When the model demonstrates a certain level of accuracy through CoT, it can significantly improve performance on challenging mathematical and logical reasoning problems. In the mathematical domain, we achieved significant improvements with different-sized and series of models. Compared with the SOTA methods, our approach can achieve stable improvements without the need for the Reward Model (RM), nor does it require fine-tuning.

## Limitations

Due to the limitations of our computing resources, we were unable to fully utilize the capabilities of Qwen2-72B-Instruct. Since applying SGR generates very long responses, the  $8 * V100$  GPU memory was insufficient to run float32, likely resulting in the lower accuracy of Qwen2-72B-Instruct than its potential. We have verified that the SGR method leads to improvements across STEM domains, but we have not yet tested whether our method can achieve similar results in more challenging AIGC tasks.

## References

- AI-MO. 2024a. [Aimo validation amc dataset on hugging face](#).
- AI-MO. 2024b. [Aimo validation dataset on hugging face](#).
- Ilias Chalkidis, Manos Fergadiotis, Dimitrios Tsarapat-sanis, Nikolaos Aletras, Ion Androutsopoulos, and Prodromos Malakasiotis. 2021. [Paragraph-level rationale extraction through regularization: A case study on european court of human rights cases](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 226–241. Association for Computational Linguistics.
- Xiaoxue Cheng, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. 2024. [Chainlm: Empowering large language models with improved chain-of-thought prompting](#). In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25 May, 2024, Torino, Italy*, pages 2969–2983. ELRA and ICCL.
- 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](#). *CoRR*, abs/2110.14168.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanbiao Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yudian Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *Preprint*, arXiv:2501.12948.
- Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. [RAFT: reward ranked finetuning for generative foundation model alignment](#). *Trans. Mach. Learn. Res.*, 2023.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Al-lonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic,

597	Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. <a href="#">The llama 3 herd of models</a> . <i>CoRR</i> , abs/2407.21783.	
613	Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. <a href="#">Complexity-based prompting for multi-step reasoning</a> . In <i>The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023</i> . OpenReview.net.	
618	Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. <a href="#">Retrieval-augmented generation for large language models: A survey</a> . <i>CoRR</i> , abs/2312.10997.	
623	Zitian Gao, Boye Niu, Xuzheng He, Haotian Xu, Hongzhang Liu, Aiwei Liu, Xuming Hu, and Lijie Wen. 2024. <a href="#">Interpretable contrastive monte carlo tree search reasoning</a> . <i>CoRR</i> , abs/2410.01707.	
627	Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024a. <a href="#">Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems</a> . In <i>Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024</i> , pages 3828–3850. Association for Computational Linguistics.	
638	Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024b. <a href="#">Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems</a> . In <i>Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024</i> , pages 3828–3850. Association for Computational Linguistics.	
649	Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. <a href="#">Measuring massive multitask language understanding</a> . In <i>9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021</i> . OpenReview.net.	
511	Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. <a href="#">Measuring mathematical problem solving with the MATH dataset</a> . In <i>Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual</i> .	655 656 657 658 659 660 661 662
511	Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. <a href="#">Training compute-optimal large language models</a> . <i>CoRR</i> , abs/2203.15556.	663 664 665 666 667 668 669 670 671
511	Ali Jarrahi, Ramin Mousa, and Leila Safari. 2023. <a href="#">SLCNN: sentence-level convolutional neural network for text classification</a> . <i>CoRR</i> , abs/2301.11696.	672 673 674
511	Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. <a href="#">Large language models are zero-shot reasoners</a> . In <i>Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022</i> .	675 676 677 678 679 680 681
511	Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. <a href="#">Let’s verify step by step</a> . In <i>The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024</i> . OpenReview.net.	682 683 684 685 686 687 688
511	Maxwell I. Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. <a href="#">Show your work: Scratchpads for intermediate computation with language models</a> . <i>CoRR</i> , abs/2112.00114.	689 690 691 692 693 694 695
511	OpenAI. 2023. Gpt-4o: Contributions. <a href="https://openai.com/gpt-4o-contributions/">https://openai.com/gpt-4o-contributions/</a> . Accessed: 2025-01-21.	696 697 698
511	OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben	699 700 701 702 703 704 705 706 707 708 709 710 711 712

713	Chess, Chester Cho, Casey Chu, Hyung Won Chung,	CJ Weinmann, Akila Welihinda, Peter Welinder, Ji-	777
714	Dave Cummings, Jeremiah Currier, Yunxing Dai,	ayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner,	778
715	Cory Decareaux, Thomas Degry, Noah Deutsch,	Clemens Winter, Samuel Wolrich, Hannah Wong,	779
716	Damien Deville, Arka Dhar, David Dohan, Steve	Lauren Workman, Sherwin Wu, Jeff Wu, Michael	780
717	Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti,	Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qim-	781
718	Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix,	ing Yuan, Wojciech Zaremba, Rowan Zellers, Chong	782
719	Simón Posada Fishman, Juston Forte, Isabella Ful-	Zhang, Marvin Zhang, Shengjia Zhao, Tianhao	783
720	ford, Leo Gao, Elie Georges, Christian Gibson, Vik	Zheng, Juntang Zhuang, William Zhuk, and Bar-	784
721	Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-	ret Zoph. 2024. <a href="#">Gpt-4 technical report</a> . <i>Preprint</i> ,	785
722	Lopes, Jonathan Gordon, Morgan Grafstein, Scott	arXiv:2303.08774.	786
723	Gray, Ryan Greene, Joshua Gross, Shixiang Shane		
724	Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris,	Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida,	787
725	Yuchen He, Mike Heaton, Johannes Heidecke, Chris	Carroll L. Wainwright, Pamela Mishkin, Chong	788
726	Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele,	Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray,	789
727	Brandon Houghton, Kenny Hsu, Shengli Hu, Xin	John Schulman, Jacob Hilton, Fraser Kelton, Luke	790
728	Hu, Joost Huizinga, Shantanu Jain, Shawn Jain,	Miller, Maddie Simens, Amanda Askell, Peter Welin-	791
729	Joanne Jang, Angela Jiang, Roger Jiang, Haozhun	der, Paul F. Christiano, Jan Leike, and Ryan Lowe.	792
730	Jin, Denny Jin, Shino Jomoto, Billie Jonn, Hee-	2022. <a href="#">Training language models to follow instruc-</a>	793
731	woo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Ka-	<a href="#">tions with human feedback</a> . In <i>Advances in Neural</i>	794
732	mali, Ingmar Kanitscheider, Nitish Shirish Keskar,	<i>Information Processing Systems 35: Annual Confer-</i>	795
733	Tabarak Khan, Logan Kilpatrick, Jong Wook Kim,	<i>ence on Neural Information Processing Systems 2022</i> ,	796
734	Christina Kim, Yongjik Kim, Jan Hendrik Kirch-	<i>NeurIPS 2022, New Orleans, LA, USA, November 28</i>	797
735	ner, Jamie Kiros, Matt Knight, Daniel Kokotajlo,	<i>- December 9, 2022.</i>	798
736	Łukasz Kondraciuk, Andrew Kondrich, Aris Kon-		
737	stantinidis, Kyle Kosic, Gretchen Krueger, Vishal	Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar,	799
738	Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan	Aafaq Iqbal khan, and Arsalan Shahid. 2024. <a href="#">The ul-</a>	800
739	Leike, Jade Leung, Daniel Levy, Chak Ming Li,	<a href="#">timate guide to fine-tuning llms from basics to break-</a>	801
740	Rachel Lim, Molly Lin, Stephanie Lin, Mateusz	<a href="#">throughs: An exhaustive review of technologies, re-</a>	802
741	Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue,	<a href="#">search, best practices, applied research challenges</a>	803
742	Anna Makanju, Kim Malfacini, Sam Manning, Todor	<a href="#">and opportunities</a> . <i>CoRR</i> , abs/2408.13296.	804
743	Markov, Yaniv Markovski, Bianca Martin, Katie		
744	Mayer, Andrew Mayne, Bob McGrew, Scott Mayer	Matthew Renze and Erhan Guven. 2024. <a href="#">Self-reflection</a>	805
745	McKinney, Christine McLeavey, Paul McMillan,	<a href="#">in LLM agents: Effects on problem-solving perfor-</a>	806
746	Jake McNeil, David Medina, Aalok Mehta, Jacob	<a href="#">mance</a> . <i>CoRR</i> , abs/2405.06682.	807
747	Menick, Luke Metz, Andrey Mishchenko, Pamela		
748	Mishkin, Vinnie Monaco, Evan Morikawa, Daniel	Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha,	808
749	Mossing, Tong Mu, Mira Murati, Oleg Murk, David	Vinija Jain, Samrat Mondal, and Aman Chadha.	809
750	Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak,	2024. <a href="#">A systematic survey of prompt engineering in</a>	810
751	Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh,	<a href="#">large language models: Techniques and applications</a> .	811
752	Long Ouyang, Cullen O’Keefe, Jakub Pachocki, Alex	<i>CoRR</i> , abs/2402.07927.	812
753	Paino, Joe Palermo, Ashley Pantuliano, Giambat-		
754	tista Parascandolo, Joel Parish, Emy Parparita, Alex	Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Ku-	813
755	Passos, Mikhail Pavlov, Andrew Peng, Adam Perel-	mar. 2024. <a href="#">Scaling LLM test-time compute optimally</a>	814
756	man, Filipe de Avila Belbute Peres, Michael Petrov,	<a href="#">can be more effective than scaling model parameters</a> .	815
757	Henrique Ponde de Oliveira Pinto, Michael, Poko-	<i>CoRR</i> , abs/2408.03314.	816
758	rny, Michelle Pokrass, Vitchyr H. Pong, Tolly Pow-		
759	ell, Alethea Power, Boris Power, Elizabeth Proehl,	Andrea Sottana, Bin Liang, Kai Zou, and Zheng Yuan.	817
760	Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh,	2023. <a href="#">Evaluation metrics in the era of GPT-4: reli-</a>	818
761	Cameron Raymond, Francis Real, Kendra Rimbach,	<a href="#">ably evaluating large language models on sequence</a>	819
762	Carl Ross, Bob Rotsted, Henri Roussez, Nick Ry-	<a href="#">to sequence tasks</a> . In <i>Proceedings of the 2023 Con-</i>	820
763	der, Mario Saltarelli, Ted Sanders, Shibani Santurkar,	<i>ference on Empirical Methods in Natural Language</i>	821
764	Girish Sastry, Heather Schmidt, David Schnurr, John	<i>Processing, EMNLP 2023, Singapore, December 6-</i>	822
765	Schulman, Daniel Selsam, Kyla Sheppard, Toki	<i>10, 2023</i> , pages 8776–8788. Association for Compu-	823
766	Sherbakov, Jessica Shieh, Sarah Shoker, Pranav	tational Linguistics.	824
767	Shyam, Szymon Sidor, Eric Sigler, Maddie Simens,		
768	Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin	Qwen Team. 2024. <a href="#">Qwq: Reflect deeply on the bound-</a>	825
769	Sokolowsky, Yang Song, Natalie Staudacher, Fe-	<a href="#">aries of the unknown</a> .	826
770	lippe Petroski Such, Natalie Summers, Ilya Sutskever,		
771	Jie Tang, Nikolas Tezak, Madeleine B. Thompson,	Hugo Touvron, Louis Martin, Kevin Stone, Peter Al-	827
772	Phil Tillet, Amin Tootoonchian, Elizabeth Tseng,	bert, Amjad Almahairi, Yasmine Babaei, Nikolay	828
773	Preston Tuggle, Nick Turley, Jerry Tworek, Juan Fe-	Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti	829
774	lippe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya,	Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-	830
775	Chelsea Voss, Carroll Wainwright, Justin Jay Wang,	Ferrer, Moya Chen, Guillem Cucurull, David Esiobu,	831
776	Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei,	Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller,	832
		Cynthia Gao, Vedanuj Goswami, Naman Goyal, An-	833
		thony Hartshorn, Saghar Hosseini, Rui Hou, Hakan	834

835	Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa,	with large language models. <i>Advances in Neural</i>	894
836	Isabel Kloumann, Artem Korenev, Punit Singh Koura,	<i>Information Processing Systems</i> , 36.	895
837	Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Di-		
838	ana Liskovich, Yinghai Lu, Yuning Mao, Xavier Mar-	Eric Zelikman, Georges Harik, Yijia Shao, Varuna	896
839	tiniet, Todor Mihaylov, Pushkar Mishra, Igor Moly-	Jayasiri, Nick Haber, and Noah D. Goodman. 2024.	897
840	bog, Yixin Nie, Andrew Poulton, Jeremy Reizen-	<a href="#">Quiet-star: Language models can teach themselves</a>	898
841	stein, Rashi Rungta, Kalyan Saladi, Alan Schelten,	<a href="#">to think before speaking</a> . <i>CoRR</i> , abs/2403.09629.	899
842	Ruan Silva, Eric Michael Smith, Ranjan Subrama-		
843	nian, Xiaoqing Ellen Tan, Binh Tang, Ross Tay-	Dan Zhang, Sining Zhoubian, Yisong Yue, Yuxiao	900
844	lor, Adina Williams, Jian Xiang Kuan, Puxin Xu,	Dong, and Jie Tang. 2024. <a href="#">Rest-mcts*: LLM self-</a>	901
845	Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan,	<a href="#">training via process reward guided tree search</a> . <i>CoRR</i> ,	902
846	Melanie Kambadur, Sharan Narang, Aurélien Ro-	abs/2406.03816.	903
847	driguez, Robert Stojnic, Sergey Edunov, and Thomas		
848	Scialom. 2023. <a href="#">Llama 2: Open foundation and fine-</a>	Qinglin Zhang, Qian Chen, Yali Li, Jiaqing Liu, and	904
849	<a href="#">tuned chat models</a> . <i>CoRR</i> , abs/2307.09288.	Wen Wang. 2021. <a href="#">Sequence model with self-adaptive</a>	905
		<a href="#">sliding window for efficient spoken document seg-</a>	906
850	Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob	<a href="#">mentation</a> . In <i>IEEE Automatic Speech Recognition</i>	907
851	Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz	<i>and Understanding Workshop, ASRU 2021, Carta-</i>	908
852	Kaiser, and Illia Polosukhin. 2017. <a href="#">Attention is all</a>	<i>gena, Colombia, December 13-17, 2021</i> , pages 411–	909
853	<a href="#">you need</a> . In <i>Advances in Neural Information Pro-</i>	418. IEEE.	910
854	<i>cessing Systems 30: Annual Conference on Neural</i>		
855	<i>Information Processing Systems 2017, December 4-9,</i>	Yifan Zhang, Jingqin Yang, Yang Yuan, and An-	911
856	<i>2017, Long Beach, CA, USA</i> , pages 5998–6008.	drew Chi-Chih Yao. 2023. <a href="#">Cumulative reasoning</a>	912
		<a href="#">with large language models</a> . <i>CoRR</i> , abs/2308.04371.	913
857	Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten		
858	Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou,	Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang,	914
859	et al. 2022. Chain-of-thought prompting elicits rea-	Xiaolei Wang, Yupeng Hou, Yingqian Min, Be-	915
860	soning in large language models. <i>Advances in neural</i>	ichen Zhang, Junjie Zhang, Zican Dong, Yifan Du,	916
861	<i>information processing systems</i> , 35:24824–24837.	Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao	917
		Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang	918
862	An Yang, Baosong Yang, Binyuan Hui, Bo Zheng,	Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen.	919
863	Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan	2023. <a href="#">A survey of large language models</a> . <i>CoRR</i> ,	920
864	Li, Dayiheng Liu, Fei Huang, Guanting Dong, Hao-	abs/2303.18223.	921
865	ran Wei, Huan Lin, Jialong Tang, Jialin Wang,		
866	Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin	Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen,	922
867	Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai,	Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny	923
868	Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-	Zhou. 2024. <a href="#">Take a step back: Evoking reasoning via</a>	924
869	qin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni,	<a href="#">abstraction in large language models</a> . In <i>The Twelfth</i>	925
870	Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize	<i>International Conference on Learning Representa-</i>	926
871	Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan,	<i>tions</i> .	927
872	Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge,		
873	Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren,	<b>A Appendix</b>	928
874	Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing	<b>A.1 Prompt</b>	929
875	Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan,	Prompt 1:	930
876	Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang,	«question»	931
877	Zhifang Guo, and Zhihao Fan. 2024a. <a href="#">Qwen2 techni-</a>	If you need to solve a current problem for a current	932
878	<a href="#">cal report</a> . <i>CoRR</i> , abs/2407.10671.	problem, what relevant knowledge do you need?	933
879	An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao,	Ask a question about relevant knowledge. Please	934
880	Bowen Yu, Chengpeng Li, Dayiheng Liu, Jian-	note: You only need to ask the question, you do	935
881	hong Tu, Jingren Zhou, Junyang Lin, Keming Lu,	not need to answer it.	936
882	Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang		937
883	Ren, and Zhenru Zhang. 2024b. <a href="#">Qwen2.5-math tech-</a>	Prompt 2:	938
884	<a href="#">nical report: Toward mathematical expert model via</a>	The answers should be short, but organized and	939
885	<a href="#">self-improvement</a> . <i>CoRR</i> , abs/2409.12122.	informative.	940
886	Haomiao Yang, Kunlan Xiang, Mengyu Ge, Hongwei	«Step Guider Question»	941
887	Li, Rongxing Lu, and Shui Yu. 2024c. <a href="#">A com-</a>		942
888	<a href="#">prehensive overview of backdoor attacks in large</a>	Prompt 3:	943
889	<a href="#">language models within communication networks</a> .	If you need to solve the current problem for the	944
890	<i>IEEE Netw.</i> , 38(6):211–218.	current step, what relevant knowledge will be	945
891	Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran,	needed in the future?	946
892	Tom Griffiths, Yuan Cao, and Karthik Narasimhan.		
893	2024. Tree of thoughts: Deliberate problem solving		

Prompt 4:

Next for the current topic to continue the next step of the answer, be sure not to repeat the previous content, to answer according to the previous content.

«Step Guidance»

## A.2 Dataset

- **MATH** The MATH dataset comprises a substantial collection of 12,500 high school-level mathematical problems, meticulously curated to cover a wide range of topics and difficulty levels. In our study, we selected the MATH dataset’s test data (5,000 problems) to evaluate our model’s performance across diverse mathematical topics and difficulty levels, ensuring a robust assessment of its generalization and problem-solving capabilities.
- **AMC23** It contains 40 data items, each including a question and an answer.
- **AIME24** The AIME24 test set is from the 2024 American Invitational Mathematics Examination. It has 30 questions, each with an answer. Among all our test sets, AIME24 is the most difficult.
- **MMLU-STEM** MMLU, or Massive Multitask Language Understanding, is a crucial benchmark for evaluating large language models. We have selected a test set with the MMLU-STEM label, which consists of a total of 3,018 problems.
- **Olympiadbench** OlympiadBench is a bilingual and multimodal scientific evaluation dataset at the Olympiad level jointly, which contains 8,952 math and physics questions from international Olympiads, Chinese Olympiads, Chinese college entrance examinations, and mock exams. We have selected a dataset with the "OE\_TO\_maths\_en\_COMP" tag, and there are a total of 675 problems in this dataset.

## A.3 A full Context of Example

Question

Paul is at a train station and is waiting for his train. He isn’t sure how long he needs to wait, but he knows that the fourth train scheduled to arrive at the station is the one he needs to get on. The first

train is scheduled to arrive in 10 minutes, and this train will stay in the station for 20 minutes. The second train is to arrive half an hour after the first train leaves the station, and this second train will stay in the station for a quarter of the amount of time that the first train stayed in the station. The third train is to arrive an hour after the second train leaves the station, and this third train is to leave the station immediately after it arrives. The fourth train will arrive 20 minutes after the third train leaves, and this is the train Paul will board. In total, how long, in minutes, will Paul wait for his train?

### • The First iteration

#### – Step Guidance Question

What is the schedule and duration of stay for each train arriving at the station before Paul’s train, and how does this affect the total waiting time for Paul’s train?

#### – Step Guidance

To answer your question, I would need more specific information about which station and which trains you are referring to. However, I can provide a general explanation of how train schedules and durations of stay can affect waiting times.

The schedule of each train arriving at a station is typically determined by the train operator and can be found on their website or at the station. The duration of stay for each train, also known as the dwell time, is the time the train spends at the station for passengers to board and disembark. This can vary depending on the number of passengers and the type of train.

If Paul’s train is scheduled to arrive after several other trains, the total waiting time for his train may be affected by the dwell times of the previous trains. If the previous trains have longer dwell times, this can cause delays for subsequent trains, including Paul’s train. However, if the previous trains have shorter dwell times and are on schedule, Paul’s train may arrive on time or even early.

Overall, the schedule and duration of stay for each train can have a significant impact on the waiting time for subsequent trains, and it’s important for train opera-

1044	tors to manage these factors to minimize	Paul needs to board. - The time Paul	1094
1045	delays and ensure on-time performance.	arrives at the station.	1095
1046	- Response for the First iteration	With this information, you can calculate	1096
1047	To calculate the total waiting time for	the total waiting time by determining the	1097
1048	Paul, we need to consider the arrival and	arrival time of the required train and sub-	1098
1049	departure times of each train and the du-	tracting the time Paul arrives at the sta-	1099
1050	ration of stay for each train.	tion.	1100
1051	1. The first train arrives in 10 minutes	- Response for the Second iteration Given	1101
1052	and stays for 20 minutes. Therefore, it	the information provided:	1102
1053	will depart 30 minutes after its arrival.	1. The first train arrives in 10 minutes	1103
1054	2. The second train arrives 30 minutes	and stays for 20 minutes, so it departs 30	1104
1055	after the first train leaves. Since the first	minutes after its arrival.	1105
1056	train leaves 30 minutes after its arrival,	2. The second train arrives 30 minutes	1106
1057	the second train arrives 60 minutes after	after the first train leaves, which is 60	1107
1058	the first train's arrival. The second train	minutes after the first train's arrival. It	1108
1059	stays for a quarter of the time the first	stays for 5 minutes (a quarter of the first	1109
1060	train stayed, which is 5 minutes (20 min-	train's stay), so it departs 65 minutes af-	1110
1061	utes / 4). Therefore, the second train will	ter the first train's arrival.	1111
1062	depart 65 minutes after the first train's		
1063	arrival.	• The third iteration	1112
1064	• The Second iteration	- Step Guidance	1113
1065	- Step Guidance	3. The third train arrives an hour after	1114
1066	3. The third train arrives an hour after	the second train leaves. Since the second	1115
1067	the second train leaves. Since the sec-	train departs 65 minutes after the first	1116
1068	ond train leaves 65 minutes after the first	train's arrival, the third train arrives 125	1117
1069	train's arrival, the third train arrives 125	minutes after the first train's arrival (65	1118
1070	minutes after the first train's arrival (65	minutes + 60 minutes). The third train	1119
1071	minutes + 60 minutes). The third train	leaves immediately after it arrives, so it	1120
1072	leaves immediately after it arrives, so it	doesn't add any additional waiting time	1121
1073	doesn't add any additional waiting time	for Paul.	1122
1074	for Paul.	4. The fourth train, which is Paul's train,	1123
1075	4. The fourth train, which is Paul's train,	arrives 20 minutes after the third train	1124
1076	arrives 20 minutes after the third train	leaves. Since the third train arrives and	1125
1077	leaves. Since the third train leaves im-	leaves immediately, the fourth train ar-	1126
1078	mediately after it arrives, the fourth train	rives 145 minutes after the first train's	1127
1079	arrives 145 minutes after the first train's	arrival (125 minutes + 20 minutes).	1128
1080	arrival (125 minutes + 20 minutes).	To find out how long Paul waits for his	1129
1081	To find out how long Paul waits for his	train, we need to calculate the time from	1130
1082	train, we need to calculate the time from	when Paul arrives at the station to when	1131
1083	when he arrives at the station to when	the fourth train arrives. Since Paul ar-	1132
1084	the fourth train arrives. Since the first	rives at the station at the same time the	1133
1085	train arrives in 10 minutes, Paul arrives	first train is scheduled to arrive (10 min-	1134
1086	at the station at this time. Therefore, Paul	utes from now), we can subtract the 10	1135
1087	waits for his train for 135 minutes (145	minutes from the 145 minutes to find	1136
1088	minutes - 10 minutes).	Paul's total waiting time.	1137
1089	In the future, to solve similar problems,	Paul's total waiting time = 145 minutes -	1138
1090	you will need to know: - The schedule of	10 minutes = 135 minutes.	1139
1091	the trains, including arrival and departure	Therefore, Paul will wait for 135 min-	1140
1092	times. - The duration of stay for each	utes for his train to arrive. The relevant	1141
1093	train. - The sequence of the trains that	knowledge needed for future steps would	1142
		be the ability to calculate time intervals,	1143

1144	understand schedules, and perform basic	arrival and stay times, and determining	1194
1145	arithmetic operations. Additionally, un-	the total waiting time for Paul based on	1195
1146	derstanding the concept of elapsed time	his arrival time and the arrival time of his	1196
1147	and how to calculate it based on given	train.	1197
1148	schedules would be beneficial.		
1149	– Response for the third iteration	– Response for the Fourth iteration	1198
1150	Continuing from the previous steps:	Given the sequence of events and the tim-	1199
1151	3. The third train arrives an hour after	ing for each train’s arrival and departure:	1200
1152	the second train leaves. Since the second	1. The first train arrives 10 minutes af-	1201
1153	train departs 65 minutes after the first	ter Paul’s arrival and stays for 20 min-	1202
1154	train’s arrival, the third train arrives 125	utes, departing 30 minutes after Paul’s	1203
1155	minutes after the first train’s arrival (65	arrival. 2. The second train arrives 30	1204
1156	minutes + 60 minutes). The third train	minutes after the first train leaves, which	1205
1157	leaves immediately after it arrives, so it	is 60 minutes after Paul’s arrival, and	1206
1158	doesn’t add any additional waiting time	stays for 5 minutes, departing 65 minutes	1207
1159	for Paul.	after Paul’s arrival. 3. The third train ar-	1208
1160	• The Forth iteration	rives an hour after the second train leaves,	1209
1161	– Step Guidance	which is 125 minutes after Paul’s arrival,	1210
1162	4. The fourth train, which is Paul’s train,	and departs immediately.	1211
1163	arrives 20 minutes after the third train		
1164	leaves. Since the third train arrives and	• The Fifth iteration	1212
1165	leaves at the same time, which is 125	– Step Guidance	1213
1166	minutes after the first train’s arrival, the	To find out how long Paul will wait for	1214
1167	fourth train arrives 145 minutes after the	his train, we need to calculate the arrival	1215
1168	first train’s arrival.	time of the fourth train and compare it	1216
1169	To find out how long Paul waits for his	to the time Paul arrived at the station.	1217
1170	train, we need to consider when he ar-	Here’s the breakdown:	1218
1171	rives at the station. Since the problem	1. **First Train:** - Arrives 10 minutes	1219
1172	doesn’t specify when Paul arrives, we	after Paul’s arrival. - Stays for 20 min-	1220
1173	can assume he arrives just before the first	utes. - Departs 30 minutes after Paul’s	1221
1174	train is scheduled to arrive, which is at	arrival.	1222
1175	the 10-minute mark. Therefore, Paul’s	2. **Second Train:** - Arrives 30 min-	1223
1176	waiting time is the time from his arrival	utes after the first train leaves, which is	1224
1177	until the fourth train arrives.	60 minutes after Paul’s arrival. - Stays	1225
1178	Given that Paul arrives just before the	for 5 minutes (a quarter of the first train’s	1226
1179	first train arrives (at the 10-minute mark)	stay). - Departs 65 minutes after Paul’s	1227
1180	and the fourth train arrives 145 minutes	arrival.	1228
1181	after the first train’s arrival, Paul’s total	3. **Third Train:** - Arrives an hour	1229
1182	waiting time is 145 minutes.	after the second train leaves. - Since	1230
1183	However, since Paul arrives 10 minutes	the second train departs 65 minutes after	1231
1184	before the first train, we need to subtract	Paul’s arrival, the third train arrives 125	1232
1185	those 10 minutes from the 145 minutes	minutes after Paul’s arrival (65 minutes +	1233
1186	to find the total time Paul actually waits	60 minutes). - Leaves immediately after	1234
1187	from his arrival. Therefore, Paul’s total	it arrives.	1235
1188	waiting time is 135 minutes (145 minutes	4. **Fourth Train:** - Arrives 20 min-	1236
1189	- 10 minutes).	utes after the third train leaves. - Since	1237
1190	In summary, the relevant knowledge	the third train arrives 125 minutes af-	1238
1191	needed to solve this problem includes	ter Paul’s arrival and leaves immediately,	1239
1192	understanding the schedule of each train,	the fourth train arrives 145 minutes after	1240
1193	calculating the departure times based on	Paul’s arrival.	1241
		Given that Paul arrived at the station at	1242
		time 0, the total waiting time for Paul is	1243

the time the fourth train arrives, which is 145 minutes. Therefore, Paul will wait for 145 minutes for his train to arrive.

This calculation takes into account the arrival and departure times of each train and the duration of stay for each train, as well as the time between the departure of one train and the arrival of the next. This information is necessary to determine the exact waiting time for Paul.

– Response for the Fifth iteration

Continuing from the established sequence:

4. The fourth train, which is the one Paul is waiting for, arrives 20 minutes after the third train leaves. Since the third train arrives 125 minutes after Paul's arrival and departs immediately, the fourth train arrives 145 minutes after Paul's arrival.

Therefore, Paul's total waiting time is 145 minutes, which is the time it takes for the fourth train to arrive after he initially arrives at the station. This total includes the waiting times for all the preceding trains to arrive, depart, and the intervals between their departures and the arrivals of the subsequent trains.

#### **A.4 Comparison with CoT and SGR**

## CoT

### Question

Paul is at a train station and is waiting for his train. He isn't sure how long he needs to wait, but he knows that the fourth train scheduled to arrive at the station is the one he needs to get on. The first train is scheduled to arrive in 10 minutes, and this train will stay in the station for 20 minutes. The second train is to arrive half an hour after the first train leaves the station, and this second train will stay in the station for a quarter of the amount of time that the first train stayed in the station. The third train is to arrive an hour after the second train leaves the station, and this third train is to leave the station immediately after it arrives. The fourth train will arrive 20 minutes after the third train leaves, and this is the train Paul will board. In total, how long, in minutes, will Paul wait for his train?

### Model directly Answer

Let's break down the schedule to determine how long Paul will have to wait.

1. **"First Train:"**
  - Arrives in 10 minutes and stays for 20 minutes.
  - Departs after 30 minutes since it arrived.
2. **"Second Train:"**
  - Arrives half an hour (30 minutes) after the first train leaves.
  - Since the first train departs at 30 minutes after it arrives (at minute 40), the second train arrives at 70 minutes.
  - It stays for a quarter of the 20-minute duration the first train stayed, which is 5 minutes.
  - So, it departs at 75 minutes.
3. **"Third Train:"**
  - Arrives an hour after the second train leaves, which is at 60 minutes + 75 minutes = 135 minutes.
  - It leaves immediately after it arrives (0 minute wait).
  - Therefore, it departs at 135 minutes.
4. **"Fourth Train (Paul's Train):"**
  - Arrives 20 minutes after the third train leaves, which is 135 minutes + 20 minutes = 155 minutes.

Since the fourth train arrives at minute 155 and Paul starts waiting at minute 0 (when the first train is scheduled to arrive in 10 minutes):

- Paul has to wait for the fourth train to arrive.
- The total time Paul will wait is 155 minutes.

Hence, Paul will wait for a total of **"155 minutes"** before his train arrives.

## SGR

### Step1

To calculate the total waiting time for Paul, we need to consider the arrival and departure times of each train and the duration of stay for each train.

1. The first train arrives in 10 minutes and stays for 20 minutes. Therefore, it will depart 30 minutes after its arrival.
2. The second train arrives 30 minutes after the first train leaves. Since the first train leaves 30 minutes after its arrival, the second train arrives 60 minutes after the first train's arrival. The second train stays for a quarter of the time the first train stayed, which is 5 minutes (20 minutes / 4). Therefore, the second train will depart 65 minutes after the first train's arrival.

### Step2

Given the information provided:

1. The first train arrives in 10 minutes and stays for 20 minutes, so it departs 30 minutes after its arrival.
2. The second train arrives 30 minutes after the first train leaves, which is 60 minutes after the first train's arrival. It stays for 5 minutes (a quarter of the first train's stay), so it departs 65 minutes after the first train's arrival.

### Step3

Continuing from the previous steps:

3. The third train arrives an hour after the second train leaves. Since the second train departs 65 minutes after the first train's arrival, the third train arrives 125 minutes after the first train's arrival (65 minutes + 60 minutes). The third train leaves immediately after it arrives, so it doesn't add any additional waiting time for Paul.

### Step4

Given the sequence of events and the timing for each train's arrival and departure:

1. The first train arrives 10 minutes after Paul's arrival and stays for 20 minutes, departing 30 minutes after Paul's arrival.
2. The second train arrives 30 minutes after the first train leaves, which is 60 minutes after Paul's arrival, and stays for 5 minutes, departing 65 minutes after Paul's arrival.
3. The third train arrives an hour after the second train leaves, which is 125 minutes after Paul's arrival, and departs immediately.

### Step5

Continuing from the established sequence:

4. The fourth train, which is the one Paul is waiting for, arrives 20 minutes after the third train leaves. Since the third train arrives 125 minutes after Paul's arrival and departs immediately, the fourth train arrives 145 minutes after Paul's arrival.

Therefore, Paul's total waiting time is 145 minutes, which is the time it takes for the fourth train to arrive after he initially arrives at the station. This total includes the waiting times for all the preceding trains to arrive, depart, and the intervals between their departures and the arrivals of the subsequent trains.

Figure 5: The comparison above shows the results of models using direct answering versus the SGR approach. The red sections in the direct answers indicate errors, while the corresponding red sections in the SGR answers are correct. Each step of the SGR-generated answer is enclosed in a box.