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

Anonymous ACL submission

Abstract

Mathematical reasoning has been a challenging aspect of 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 require massive inference datasets as training datasets or rely on few-shot methods that often sacrifice accuracy. To address this bottleneck in mathematical reasoning, we propose a novel method called Step Guidance Reasoning without involving further model finetuning. In this approach, LLMs reflect on small reasoning steps – similar to how humans deliberate on 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. Our method significantly improved the math performance, raising the accuracy on the AMC23 dataset from 30% to 57.5%, a relative improvement of 91.7%, and on the sampled level 5 problem of the MATH dataset, we achieved a relative accuracy improvement of 55.8%, increasing from 43% to 67%.

1 Introduction

Since the introduction of Chain-of-Thought (COT) (Wei et al., 2022) reasoning on LLMs (Naveed et al., 2023; Zhao et al., 2023), complex problems can be broken down into smaller but more manageable reasoning steps during LLMs inference, much like how humans approach logical reasoning. However, complex mathematical reasoning remains as a significant challenge for LLMs (He et al., 2024). 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 becomes the bottleneck (Hoffmann et al., 2022).

To tackle these, on one hand, cumulative reasoning (Zhang et al., 2023) has been proposed as a new method, which has made great improvements on MATH datasets. Cumulative reasoning significantly enhances problem-solving by decomposing the task into smaller, more manageable parts and utilizing previous propositions for effective combination. On the other hand, Zheng et al. proposed a “Take one Step Back” method, which introduced concepts and principles to guide the model reasoning using results from high-level descriptions of the 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, scaling the test-time computation (Snell et al., 2024).

However, both methods showed improvement on medium-difficulty datasets such as the MATH dataset (Hendrycks et al., 2021b) and MMLU Stem (Hendrycks et al., 2021a), but when coming to challenging math problems, such as competition level math (AI-MO, 2024b,a), neither of them work well. To be more specific, we observed that more challenging math tasks often require more thought reasoning steps (see analysis of Figure 2) to complete the answer. Inspired by these observations, we propose a method called Step Guidance Reasoning (SGR) that introduces guidance thought in step-by-step reasoning, and SGR can improve challenging math problems without the finetuning model. In more detail, in each reasoning step of our approach, the model is prompted to firstly self-question what to do next, secondly to self-answer this question that can help next-step generation, and lastly to use this reflection to guide the subsequent generation process. Through this method, we observe significant improvements in solving complex problems, particularly in multi-step tasks like the Olympic Mathematical Challenge, provided the model already demonstrates a certain level of accuracy on

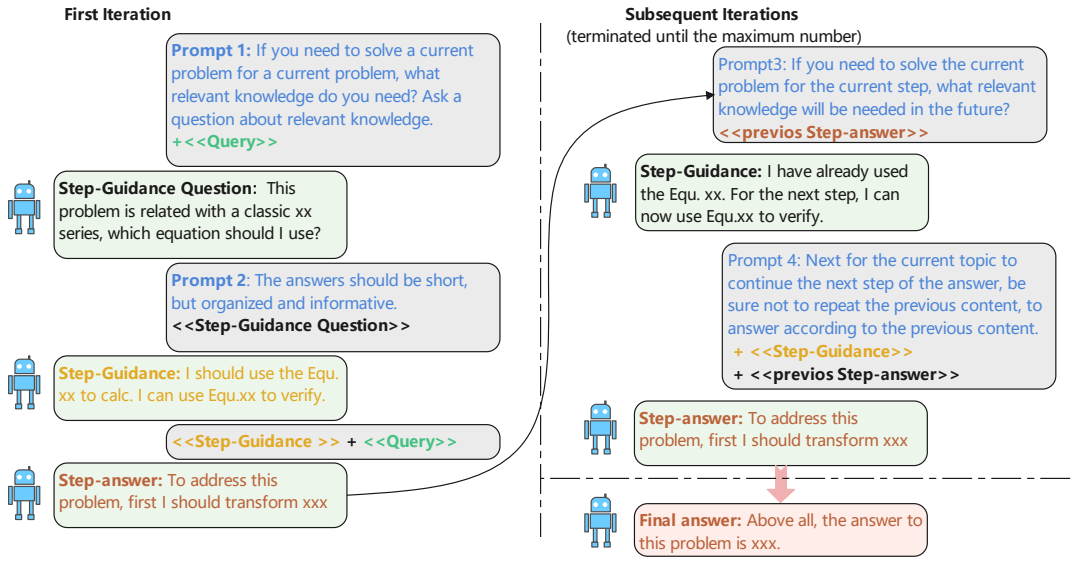

Figure 1: Illustration of how our proposed SGR method generates Guidance and reasoning for each step.

the dataset. We went from 40 percent accuracy to 65 percent accuracy in AMC23 and from 6.7 percent accuracy to 13 percent accuracy on the AIME24 test set. Increased from 43% to 67% on our sampled MATH dataset.

To summarise our contributions, we propose a method that enables the model to determine the next steps without requiring fine-tuning in the domain of mathematical logic, leading to substantial improvements on challenging math datasets.

2 Method

Our SGR involves multiple thought steps during the inference time, each step consisting of two main generation components, a step guidance generation and a reasoning generation.[1] The overall illustration of our method is shown in Figure 1, the model first generates a step guidance related to the given math problem and then uses the generated step guidance to make a one-step answer at the current step. This procedure terminates until the model generates the final answer or reaches the budget of iterations.

2.1 Step Definition

Since our method is based on an instruction model and does not require further finetuning, we describe how we define one step during inference. Theoretically, a step can be defined using different granularities, such as token level (Zelikman et al., 2024), sentence level (Jarrahi et al., 2023), paragraph level (Chalkidis et al., 2021), or block level annotated by human experts (Lightman et al., 2023). Considering SGR involves no finetuning and challenging math problems often require thousands-of-tokens answers (Fu et al., 2022), we choose to use one paragraph for one step.

In detail, we found that for English outputs, ".\n\n" can serve as an effective delimiter for logical inference for most instruction/chat models, such as GPT-4 and Qwen. However, directly splitting the reasoning at every ".\n\n" can cause repetitions in the model's generation that instead of progressing to the next step, the model tends to repeatedly analyze the first step, mistaking it as an instruction to reanalyze the problem. To mitigate this problem, we introduced a step length constraint, that each step, indicated by ".\n\n", require a minimum number of tokens.[2] Theoretically, each problem could require different step lengths, and even within one problem, different steps may vary in length. Ideally, we would finetune the LLMs over manually labelled data with a special step token to explicitly distinguish between steps, however, we do not consider this in this paper.

2.2 Step Guidance

The process of generating the step guidance involves two parts. For the first iteration, the LLM is asked the first step guidance question additionally, which is related to the query and includes its relevant knowledge. Then the LLM answers

[1] All used prompts are listed in Appendix A.1.

[2] While this mitigated some of the repetition, LLMs still exhibited long repetitive patterns in subsequent steps, a problem that cannot be entirely solved without fine-tuning.

Model	MATH	AMC23	AIME24
Qwen2-Math-72B-instruct	34	60	**20**
Qwen2.5-Math-7B-instruct	57	62.5	16.7
GPT4 (2024/10/15)	30	27.5	6
Qwen2-72B-instruct			
w. CoT (`top_p=1.0`)	40	30	0
w. SGR (`top_p=1.0`)	53	57.5	3
w. CoT (`top_p=0.7`)	43	40	6.7
w. SGR (`top_p=0.7`)	**67**	**65**	13

Table 1: Accuracy (%) of different models over MATH, AMC23, and AIME24 datasets. The best results are in **Bold** for each dataset. The AMC23 and AIME24 results for Qwen2-Math-72B-instruct and Qwen2.5-Math-7B-instruct are taken from the Qwen2-Math (Yang et al., 2024a) and Qwen2.5-Math (Yang et al., 2024c) reports, respectively.

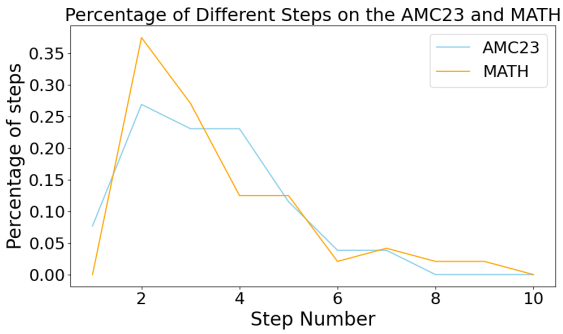

Figure 2: It shows the fraction of the first steps in which the model had the correct answer for the correct result on the MATH dataset.

this question independently, leading to the generated step guidance. For subsequent iterations, the LLM directly answers what relevant knowledge is needed to proceed with the next step. All generated step guidances are used only for the current step and are not retained for subsequent steps.

2.3 Reasoning

During the reasoning stage, the step guidance and the previously accumulated reasoning results are included in the prompt to facilitate continued reasoning, which is influenced by the step guidance's problem-solving format and strategies. The generation process is halted once it reaches the token ".\n\n" with a minimum length, marking the completion of the current step. This approach resembles Reach-Augmented Generation (Gao et al., 2023), but the additional context is generated by the LLM itself rather than being pre-existing.

The above Step Guidance and Reasoning procedures terminate

3 Experiments

3.1 Experimental Setups

Dataset For evaluation, we use three representative challenging math benchmarks, AMC23 (), level 5 test cases of MATH (Hendrycks et al., 2021b),[3] and AIME24 (AI-MO, 2024b). Among AIME24, AMC23 and level 5 of MATH, AIME24 is the most challenging while level 5 of MATH is relatively easy. All three datasets provide the problem as a query and a reference answer, and we report the accuracy by verifying LLM's final generation with the reference answer. Due to different difficulties and evaluation efficiency, we evaluate the AIME24 and AMC23 test data manually, and we deploy an automatic evaluation tool, GPT4, to compare the generated answer and the reference answer for the MATH benchmark. We used 8*V100 for the experiment and each problem in the test dataset is about 2 to 10 minutes in the inference stage.

Hyperparamters Given that SGR necessitates a model with strong comprehensive abilities, We choose Qwen2-72B-instruct (Yang et al., 2024b) as our experimental model. For the decoding strategy, we use temperature as 1 and set topp to 1 and 0.7 for sampling.[4] The step length for MATH was specified as 300, while for the AIME24 dataset, it was set to 500. We use a maximum of 10 reasoning steps for all datasets. All experiments are run 3 times, and we report the average accuracy.

3.2 Experiment Results

Table 1 shows the overall results of the proposed methods. The results show that for both top_p setups, SGR consistently improves the performance over three datasets compared with standard CoT. With top_p equals 0.7, the relative improvement is 55.8% for the MATH test set, 62.5% for AMC23, and 94.0%. Compared with sota math-specific open-sourced models, Qwen2-Math-72B-instruct and Qwen2.5-Math-7B, we achieve comparable performance, while our SGR do not require fine-tuning data.

[3] For evaluation efficiency, we sampled 100 instances with level-5 label. We observed consistent results using the full MATH benchmarks.

[4] We observed that top_p decoding tends to mitigate repetition compared with greedy decoding.

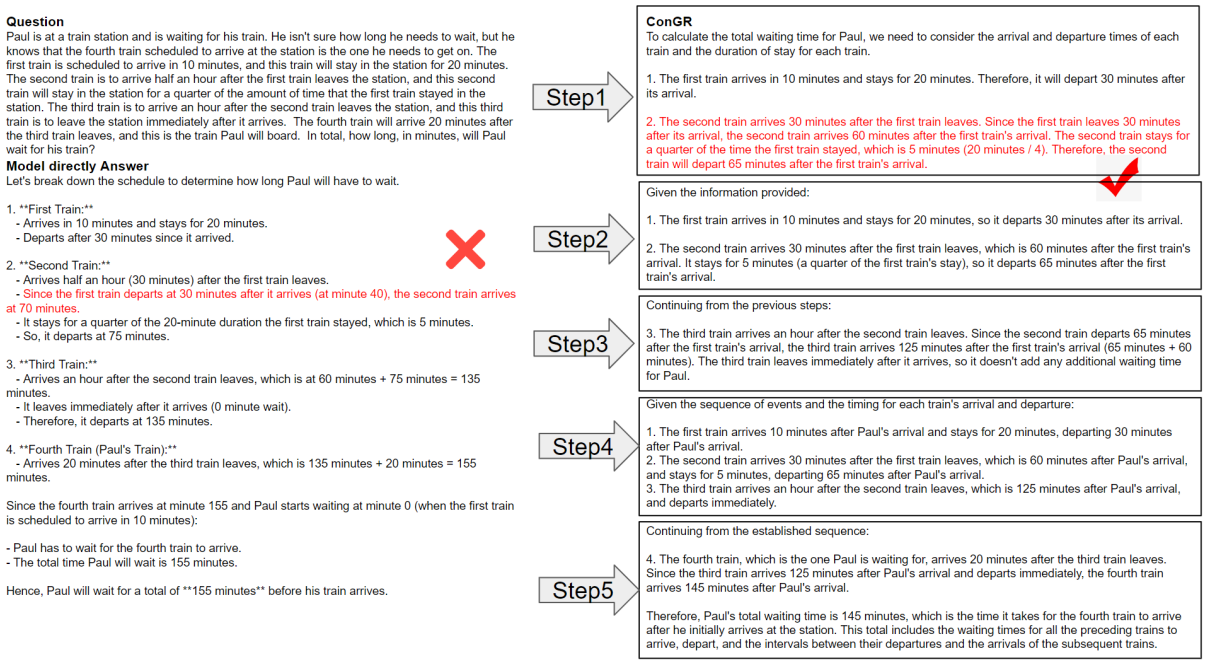

Figure 3: The comparison above shows the results of models using direct answering versus the ConGR approach. The red sections in the direct answers indicate errors, while the corresponding red sections in the SGR answers are correct. Each step of the SGR-generated answer is enclosed in a box.

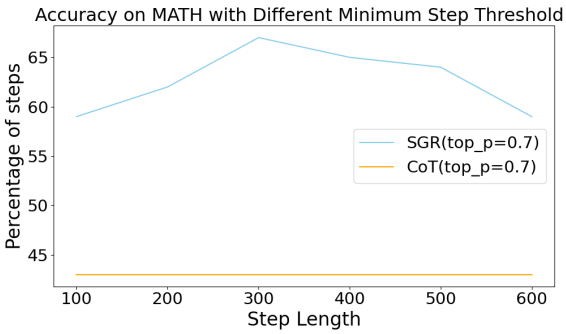

Figure 4: The line chart shows the change in accuracy on the MATH dataset using different length thresholds

3.3 Analysis

Answer Step We first analyse the number of steps needed to ask a math problem across different datasets (see Figure 2). The number of steps in MATH is more concentrated in step 2 (37%), while answering AMC23 requires more steps, concentrated in step 3 and step 4 (46% in total).

Step Length We experiment with the model performance with different step lengths, ranging from 100 to 600, over the MATH dataset. As illustrated in Figure 4, we observed that the accuracy improves as the step length increases from 59% at 100 to a peak of 67% at length 300. However, when step length further increases, the accuracy begins to fluctuate, and eventually reverting to 59% at 600 steps. Compared with standard CoT methods, SGR consistently outperforms CoT. This highlights the importance of identifying an optimal step length to achieve maximum accuracy and demonstrate the robustness of our method using different step lengths.

Case Study To understand how our method improves the reasoning procedure, we demonstrate an example in Figure 3. Compared to standard CoT and SGR 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 direct CoT reasoning makes an error. The full contents of this example are included in the Appendix A.2.

4 Conclusion

We propose a step-by-step reasoning method incorporating guidance generation within each step for math problem tasks. Our method can be applied to a general instruction-finetuned model and does not require further fine-tuning. We demonstrate that SGR approach gains absolute accuracy improvement over MATH (level 5 problems), AMC23 and AIME24 benchmarks by 24%, 25% and 6.3% respectively, using qwen2-72B-instruct.

Limitations

Due to the limitation of our calculation resource in the inference stage, the test data in the experiment can only be sampled from a part of level 5 of MATH. In addition, the GPT4 evaluation results are limited in scope. Therefore, we used manual evaluation methods on the AMC23 and the AIME test datasets.

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 Tsarapatsanis, Nikolaos Aletras, Ion Androutsopoulos, and Prodromos Malakasiotis. 2021. Paragraph-level rationale extraction through regularization: A case study on european court of human rights cases. *Preprint*, arXiv:2103.13084.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *ArXiv preprint*, abs/2402.14008.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

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

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *ArXiv preprint*, abs/2203.15556.

Ali Jarrahi, Ramin Mousa, and Leila Safari. 2023. Slcnn: Sentence-level convolutional neural network for text classification. *Preprint*, arXiv:2301.11696.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *ArXiv*, abs/2305.20050.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. 2021. Show your work: Scratchpads for intermediate computation with language models. *ArXiv preprint*, abs/2112.00114.

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.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024a. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024b. Qwen2 technical report. *ArXiv preprint*, abs/2407.10671.

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

E. Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. 2024.

Quiet-star: Language models can teach themselves to think before speaking. *ArXiv*, abs/2403.09629.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. 2023. Cumulative reasoning with large language models. *ArXiv preprint*, abs/2308.04371.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *ArXiv preprint*, abs/2303.18223.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H Chi, Quoc V Le, and Denny Zhou. 2023. Take a step back: Evoking reasoning via abstraction in large language models. *ArXiv preprint*, abs/2310.06117.

A Appendix

A.1 Prompt

Prompt 1:
«question»
If you need to solve a current problem for a current problem, what relevant knowledge do you need? Ask a question about relevant knowledge. Please note: You only need to ask the question, you do not need to answer it.

Prompt 2:
The answers should be short, but organized and informative.
«Step Guider Question»

Prompt 3:
If you need to solve the current problem for the current step, what relevant knowledge will be needed in the future?

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 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 operators to manage these factors to minimize delays and ensure on-time performance.

- Response for the First iteration

  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.

- The Second iteration

  - Step Guidance

    3. The third train arrives an hour after the second train leaves. Since the second train leaves 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.

    4. The fourth train, which is Paul's train, arrives 20 minutes after the third train leaves. Since the third train leaves immediately after it arrives, the fourth train arrives 145 minutes after the first train's arrival (125 minutes + 20 minutes).

    To find out how long Paul waits for his train, we need to calculate the time from when he arrives at the station to when the fourth train arrives. Since the first train arrives in 10 minutes, Paul arrives at the station at this time. Therefore, Paul waits for his train for 135 minutes (145 minutes - 10 minutes).

    In the future, to solve similar problems, you will need to know: - The schedule of the trains, including arrival and departure times. - The duration of stay for each train. - The sequence of the trains that Paul needs to board. - The time Paul arrives at the station.

    With this information, you can calculate the total waiting time by determining the arrival time of the required train and subtracting the time Paul arrives at the station.

  - Response for the Second iteration 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.

- The third iteration

  - Step Guidance

    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.

    4. The fourth train, which is Paul's train, arrives 20 minutes after the third train leaves. Since the third train arrives and leaves immediately, the fourth train arrives 145 minutes after the first train's arrival (125 minutes + 20 minutes).

    To find out how long Paul waits for his train, we need to calculate the time from when Paul arrives at the station to when the fourth train arrives. Since Paul arrives at the station at the same time the first train is scheduled to arrive (10 minutes from now), we can subtract the 10 minutes from the 145 minutes to find Paul's total waiting time.

    Paul's total waiting time = 145 minutes - 10 minutes = 135 minutes.

    Therefore, Paul will wait for 135 minutes for his train to arrive. The relevant knowledge needed for future steps would be the ability to calculate time intervals, understand schedules, and perform basic arithmetic operations. Additionally, un-

derstanding the concept of elapsed time and how to calculate it based on given schedules would be beneficial.

  - Response for the third iteration

    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.

- The Forth iteration
  - Step Guidance

    4. The fourth train, which is Paul's train, arrives 20 minutes after the third train leaves. Since the third train arrives and leaves at the same time, which is 125 minutes after the first train's arrival, the fourth train arrives 145 minutes after the first train's arrival.

    To find out how long Paul waits for his train, we need to consider when he arrives at the station. Since the problem doesn't specify when Paul arrives, we can assume he arrives just before the first train is scheduled to arrive, which is at the 10-minute mark. Therefore, Paul's waiting time is the time from his arrival until the fourth train arrives.

    Given that Paul arrives just before the first train arrives (at the 10-minute mark) and the fourth train arrives 145 minutes after the first train's arrival, Paul's total waiting time is 145 minutes.

    However, since Paul arrives 10 minutes before the first train, we need to subtract those 10 minutes from the 145 minutes to find the total time Paul actually waits from his arrival. Therefore, Paul's total waiting time is 135 minutes (145 minutes - 10 minutes).

    In summary, the relevant knowledge needed to solve this problem includes understanding the schedule of each train, calculating the departure times based on arrival and stay times, and determining the total waiting time for Paul based on his arrival time and the arrival time of his train.

  - Response for the Fourth iteration

    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.

- The Fifth iteration
  - Step Guidance

    To find out how long Paul will wait for his train, we need to calculate the arrival time of the fourth train and compare it to the time Paul arrived at the station. Here's the breakdown:

    1. **First Train:** - Arrives 10 minutes after Paul's arrival. - Stays for 20 minutes. - Departs 30 minutes after Paul's arrival.

    2. **Second Train:** - Arrives 30 minutes after the first train leaves, which is 60 minutes after Paul's arrival. - Stays for 5 minutes (a quarter of the first train's stay). - Departs 65 minutes after Paul's arrival.

    3. **Third Train:** - Arrives an hour after the second train leaves. - Since the second train departs 65 minutes after Paul's arrival, the third train arrives 125 minutes after Paul's arrival (65 minutes + 60 minutes). - Leaves immediately after it arrives.

    4. **Fourth Train:** - Arrives 20 minutes after the third train leaves. - Since the third train arrives 125 minutes after Paul's arrival and leaves immediately, the fourth train arrives 145 minutes after Paul's arrival.

    Given that Paul arrived at the station at time 0, the total waiting time for Paul is 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.