

## A Implementation Details

### A.1 Inference

**Inference from a sequential reasoning model.** To generate responses from sequential reasoning models, such as DeepSeek-R1-Distill-7B and the RFT model, we use the prompt provided below. The same prompt was used for fine-tuning DeepSeek-R1-Distill-7B to derive the RFT model. During inference, the question is appended to the prompt and the model is called in the completions format. Following guidelines suggested by DeepSeek [3], we set the generation temperature to 0.6 to mitigate repetitive outputs. Additionally, we enforce a maximum token limit of 36,000 per response, truncating any outputs exceeding this threshold.

#### Sequential Reasoning Prompt

Your role as an assistant involves thoroughly exploring questions through a systematic long thinking process before providing the final precise and accurate solutions. This requires engaging in a comprehensive cycle of analysis, summarizing, exploration, reassessment, reflection, backtracking, and iteration to develop well-considered thinking process.

Please structure your response into two main sections: **Thought** and **Solution**.

- In the Thought section, detail your reasoning process using the specified format: `<think>` {thought with steps separated with "`\n \n`"} `</think>` Each step should include detailed considerations such as analyzing questions, summarizing relevant findings, brainstorming new ideas, verifying the accuracy of the current steps, refining any errors, and revisiting previous steps.
- In the Solution section, based on various attempts, explorations, and reflections from the Thought section, systematically present the final solution that you deem correct. The solution should remain a logical, accurate, concise expression style and detail necessary step needed to reach the conclusion.

Now, try to solve the following question through the above guidelines. Return your final response within `\boxed{}`.

**SPRINT Inference.** During inference, we use the following prompt to guide the generation of plans and executions from the SPRINT model. Although the model is fine-tuned to produce an entire trajectory—including all plans and executions—in a single generation, we manage model invocations and output token handling to alternate between planner and executor roles effectively.

To restrict the model’s outputs to either a single plan or execution per invocation, we employ specific stop tokens. Generation is terminated once the model produces any of the following strings, indicating the completion of a plan or execution segment: `{</Execution_, </Plan_, </Final_answer>, </execution_}`.

When generating a plan for round  $i$ , we feed the prompt along with the input query and the cumulative context, which includes all preceding plans and executions, to the SPRINT model. Conversely, to generate the execution corresponding to a particular `prompt_i.j`, we provide the model with the prompt, the input query, all previously generated plans and executions up to round  $i - 1$ , and the text from plan  $i$  until the end of `prompt_i.j`. This structured context management allows us to reuse the same prompt for both planning and execution tasks seamlessly.

The model is permitted a maximum of 12 stages to produce a final answer. To enforce this constraint, we append `"<Final_answer>\n"` at the end of the prompt when invoking the model at the 12th stage. Generated responses for each plan or execution are limited to 8,000 tokens, with any excess tokens truncated accordingly. This prompt is identical to that used during model fine-tuning.

### SPRINT Prompt

You are an AI system that follows a systematic long thinking process to arrive at the precise and accurate answer to the below math question specified within `<Question>` and `</Question>` tags. The solution is generated over multiple phases, where each phase consists of a **plan** and an **execution**.

**Planning.** At phase  $p$ , you must first create a plan within `<Plan_p>` and `</Plan_p>` tags by thinking out loud and planning the tasks that need to be executed next.

- Your plan may involve detailed considerations such as analyzing the question, summarizing relevant findings, brainstorming new/alternative approaches, verifying the accuracy of the current steps, refining any errors, or revisiting previous steps.
- Since you may think about multiple aspects of the problem within the same plan, you must insert a line break using `"- - - -"` before you transition from one train of thought to another.
- While generating the plan, if you identify a task that needs to be executed, you must create a prompt that clearly specifies the task within `<prompt_p.k>` and `</prompt_p.k>` tags where  $k$  starts from 1.
- When planning within each phase, you must create prompts that can be run independent of the results from the other prompts, to achieve speedup through parallelization. You must also try to minimize the number of phases required further to arrive at the accurate final answer.

**Execution.** After creating the plan, you must carry out the tasks identified in the plan sequentially within `<Execution_p>` and `</Execution_p>` tags. For the prompt labeled as `prompt_p.k`, the corresponding execution must be generated within `<execution_p.k>` and `</execution_p.k>` tags.

If the plans and execution results you receive are sufficient to generate the accurate final answer, you must respond with the final answer within `<Final_answer>` and `</Final_answer>` tags. The numerical answer must be within `\boxed{\}`.

788

## 789 A.2 Dataset Curation

790 **Step Extraction.** For step extraction, we use the GPT-4o model with the temperature set to 0. The  
791 following prompt is used to extract steps from a reasoning trajectory generated by DeepSeek R1. In  
792 the prompt, we use the term "Component" to refer to the extracted steps to prevent the model from  
793 confusing it with the traditional use of the term "step" in a math solution which could be a single  
794 operation as opposed to a logical part of the solution. Components, as defined here, may involve tasks  
795 such as identifying subsequent actions, validating previous results, proposing alternative methods, or  
796 comparing solutions derived through different strategies. For each component, the model starts by  
797 thinking out loud about what needs to be done and then carries out the identified task. We refer to the  
798 first part as the plan and the second as the execution and extract them separately using this prompt.

799 The reasoning trajectory passed to the model as input is formatted by labeling each line/sentence with  
800 a unique line number and the model provides the range of line numbers for each plan and execution  
801 within a component. This minimizes the number of output tokens that have to be generated by the  
802 model, consequently reducing costs. The line numbers are later parsed from the response to infer the  
803 block of text that is relevant to each plan or execution.

### Step Extraction Prompt

Given below is a math problem and a well-thought out solution to the problem generated by an AI model. The solution contains multiple components (progressing with next steps, verifying past steps, proposing alternative methods, comparing solutions across different methods, etc.). Within each component, there are three phases:

804

- **Planning:** Here, the model first thinks out loud and plans what it needs to do.
- **Execution:** Here, the model follows the plan and executes it.
- **Commenting:** Here, the model comments on the execution results with phrases such as "Yes, that seems right", "Both methods lead to the same answer, etc."

Note that the verification of an execution should be considered as a separate component and not as the commenting phase of the same component.

I am building a new AI system to solve such math problems. This system will consist of two separate AI models – a planner and an executor.

- **Planner:** The planner will receive all the components of the solution completed so far and will need to think aloud and generate a plan for the next component. Then, it needs to provide a prompt to the executor model to execute a specific task.
- **Executor:** The executor will receive all the components of the solution completed so far, the plan for the next component generated by the planner, and the prompt generated by the planner. It will need to execute the specified task.

To train these two AI models, I must generate training data by breaking down the solution provided below into individual components. For each component, clearly provide the following details:

**Required Response Format:**

### Component X (Line Number Range)

- **Description:** Brief explanation of what this component achieves.
- **Plan:** Lines (minimal number of lines to describe the plan clearly).
- **Prompt:** A precise, actionable instruction for the executor based explicitly on the above plan.
- **Execution:** Lines (specific line numbers performing the planned task).
- **Comment:** Lines (reflective comments or **Lines not found** if missing).

**Important Notes:**

- The planning phase should only include a minimal number of lines required to specify what needs to be done. The remaining lines from the component where the model carries out the plan should be included in the execution phase.
- There MUST be NO overlap between the line numbers of different components.
- There MUST be NO overlap between the line numbers of the planning and execution phases of the same component.
- All the lines in the solution should be covered by the components.
- Use the line number mentioned at the start and end of each line to identify the line when specifying the line number range.
- The prompt to the executor model must be a very specific instruction that the executor can follow to complete the required task. The executor must not perform more tasks than required. The prompt can refer to the plan for that component by saying "the above plan".
- If the model does not comment on the execution results within a component, the corresponding bullet point can be written as **Comment:** Lines not found

806 **DAG Creation.** For DAG creation, we use the GPT-4o-mini model with the temperature set to  
807 0. The following prompt is used to infer the DAG in the form of a parent dictionary, where each  
808 key refers to a step extracted above and the corresponding values refer to the steps on which the key  
809 depends.

#### DAG Creation Prompt

Given below is a well-thought out solution to a math problem generated by an AI system. The system consists of a **planner** and an **executor**. The planner model thinks out loud and plans the next component of the problem solution. Then, it provides a prompt along with the plan to an executor model. The executor then follows the instructions in the prompt and uses context from the plan to carry out the given task.

The solution consists of multiple **components**, each containing the following:

- **Description:** A brief description of what the component does.
- **Plan:** The plan generated by the planner.
- **Prompt:** Instructions generated by the planner enclosed within <prompt> tags.
- **Execution:** Output provided by the executor.

Though the executions are run sequentially in this solution, some of the executions may be parallelized to improve speed. Identify and explain which components can run in parallel and determine the best way to parallelize them to maximize speed. Note that parallel runs should not have co-dependency.

The parallelization schedule can be represented as a **directed acyclic graph (DAG)** where the nodes are the component numbers. You need to represent the DAG as a parent dictionary where each node is a key and its value is a list of nodes that point to it, i.e., the nodes that must be executed immediately before it. For a key node, do not include any nodes in its value that can be run in parallel with it.

#### Format of parent dictionary:

Let us consider a simple example. Suppose that the following constraints hold:

- Component 1 needs to be run before any other component
- Components 2, 3, 4 can be run in parallel after 1
- Component 5 which depends on the results of 2 and 3 can be run after 2 and 3
- Component 6 which depends on the results of 4 and 5 can be run after 4 and 5

The parent dictionary for this example *\*MUST\** be represented as a python dictionary as follows:

```
parent_dictionary = {  
    1: [],  
    2: [1],  
    3: [1],  
    4: [1],  
    5: [2, 3],  
    6: [4, 5]  
}
```

810

811 Using the resulting DAG, we can reorganize the components into interleaved plans and executions to  
812 obtain a parallelizable reasoning trajectory. A simple strategy involves assigning components at the  
813 same DAG depth to the same planning-execution rounds. However, further optimization can reduce  
814 the total number of rounds required to reach the final answer.

815 **Packing.** The objective of packing is to optimally assign round numbers to each component. To  
816 achieve this, we apply the following greedy heuristics:

- If a component’s execution consists of fewer than three lines, it is merged directly with its corresponding plan. This approach reduces overhead from additional prompt writing and executor invocation. Through fine-tuning on trajectories with merged short executions, the planner learns to carry out short or trivial executions on its own.
- If a component  $C$  depends on a plan-only component  $P$ , then  $C$ ’s plan is independent of the execution results from  $P$ ’s round. When all of  $C$ ’s parent components satisfy this condition,  $C$  is merged into the same planning round as  $P$  by combining their respective plans.

As a result, we obtain optimal round numbers for each component which can then be used for generating the fine-tuning trajectory.

### A.3 Fine-tuning

We conducted supervised fine-tuning (SFT) of our models by training on the reasoning trajectories. Initially, we experimented with more efficient fine-tuning techniques such as LoRA[34] and qLoRA [35]. However, since LoRA did not adequately enable the models to adhere to the desired response format, we proceed with full fine-tuning instead.

Fine-tuning was primarily executed on a single machine with eight NVIDIA A100 GPUs with 40 GB memory per GPU. We use the `ms-swift` framework [36], a fine-tuning toolkit provided by the Modelscope community.

Each model is fine-tuned for 5 epochs. Due to the long-context required for reasoning traces and the memory constraints, we use a batch size of 1 during the training. We use `bfloat16` precision, an initial learning rate of  $1 \times 10^{-5}$ , and a weight decay factor of  $1 \times 10^{-4}$ . The learning rate scheduling consists of a linear warm-up phase during the first 5% of training steps, subsequently followed by linear decay to zero over the remaining training iterations. Model evaluation is conducted every 100 steps, and the best-performing model based on evaluation loss is retained.

To optimize memory usage during training, we integrate several efficiency strategies, notably the DeepSpeed ZeRO Redundancy Optimizer [37, 38] and 4-bit quantization. DeepSpeed’s ZeRO optimizer offers a set of memory-partitioning strategies that trade off memory savings against communication overhead. In many workloads, ZeRO Stage 1 or 2 strikes the best balance between memory efficiency and communication cost; however, since we need to train on long sequences, our per-GPU memory demands exceed what those stages can support. Therefore, we adopted ZeRO Stage 3 to train with extended context lengths without OOM errors.

### A.4 Evaluation

For model evaluation, we leverage vLLM [39] to serve our models. Specifically, each 7B-scale model (SPRINT, RFT, and DeepSeek-R1-Distill-7B) is deployed on a single NVIDIA A100 GPU with 40 GB of memory.

To enhance evaluation accuracy, we instruct the models to encapsulate their final answers within `\boxed{\}`. For evaluations on the MATH500 and Countdown tasks, we leverage the Math-Verify library alongside SymPy for equivalence checking, ensuring robustness against mathematically equivalent but differently expressed solutions. In the GPQA task, accuracy is determined by comparing explicitly generated option labels (e.g., A, B, C, D) directly with the corresponding ground-truth options.

Despite providing explicit formatting instructions, we occasionally observe deviations by the models from the specified output format. For instance, during the Countdown task evaluation, the models occasionally produce outputs in unexpected formats (e.g., `\times`, `\times`), complicating the parsing procedure. To mitigate these issues, we incorporate the GPT-4o-mini model as an auxiliary evaluator to judge the generated answer against the ground truth.

### A.5 Baselines

Below we present the implementation details for the baselines used in our study:

- **SoT (Skeleton of Thought):** For SoT-chat, we use the Qwen2.5-7B-Instruct model with the temperature set to 0.1. For SoT-reasoning, we use the DeepSeek-R1-Distill-Qwen-7B

866 model with the temperature set to 0.6. We use the same prompts as reported in the original  
867 paper [25]. After the full solution is generated, we send it to GPT-4o-mini to extract the  
868 final answer for evaluation.

- 869 • **RFT (Reasoning Fine-tuned Model):** This baseline was created by fine-tuning the  
870 DeepSeek model on sequential R1 trajectories. During inference, we set the temperature to  
871 0.6 and employed the sequential reasoning prompt provided in Appendix A.1.
- 872 • **Parallel Sampling + Self-consistency:** The Qwen2.5-7B-Instruct model was used with a  
873 temperature of 0.3, generating 20 independent samples per problem. The majority answer is  
874 considered as the predicted answer for the evaluation of this baseline.

## 875 B Sample Demonstrations

### 876 B.1 Demonstration of the Data Curation Pipeline

877 To illustrate our data curation pipeline, we examine a representative example from the MATH training  
878 dataset. Consider the following question:

879 **Question:** The vertex of the parabola described by the equation  $y = -3x^2 - 30x - 81$  is  $(m, n)$ .  
880 What is  $n$ ?

881 Next, we provide the response generated by DeepSeek-R1 for this question with each logical  
882 component in a separate box. Note that the original response does not provide any separation among  
883 its logical components and the full text is generated as a single long trajectory.

#### Outline of DeepSeek-R1 response

Okay, so I need to find the vertex of the parabola given by the equation  $y = -3x^2 - 30x - 81$ , and specifically determine the  $n$ -coordinate of the vertex, which is  $(m, n)$ . Hmm, let's start by recalling how to find the vertex of a quadratic equation. I remember that quadratic equations are in the form  $y = ax^2 + bx + c$ , and their graphs are parabolas. The vertex of a parabola can be found using the formula for the  $x$ -coordinate, which is  $-b/(2a)$ . Then once we have the  $x$ -coordinate, we can plug that back into the original equation to find the corresponding  $y$ -coordinate, which would be  $n$  in this case.

So let's identify  $a$ ,  $b$ , and  $c$  from the given equation. Comparing  $y = -3x^2 - 30x - 81$  with the standard form  $y = ax^2 + bx + c$ , it looks like  $a = -3$ ,  $b = -30$ , and  $c = -81$ .

Wait, is that right? Let me check again. The coefficient of  $x^2$  is  $-3$ , so  $a = -3$ . The coefficient of  $x$  is  $-30$ , so  $b = -30$ . The constant term is  $-81$ , so  $c = -81$ . Okay, that seems correct.

Now, using the formula for the  $x$ -coordinate of the vertex:  $m = -b/(2a)$ . Plugging in the values of  $a$  and  $b$ , we get  $m = -(-30)/(2 * (-3))$ . Let's compute that step by step. The numerator is  $-(-30)$ , which is positive 30. The denominator is 2 times  $a$ , which is  $2 * (-3) = -6$ . So  $m = 30/(-6) = -5$ . Therefore, the  $x$ -coordinate of the vertex is  $-5$ . That seems straightforward.

Now, to find the  $y$ -coordinate  $n$ , I need to substitute  $x = -5$  into the original equation. Let's do that. The original equation is  $y = -3x^2 - 30x - 81$ . Plugging in  $x = -5$ :

$y = -3 * (-5)^2 - 30 * (-5) - 81$ . Let's compute each term step by step. First,  $(-5)^2$  is 25, so  $-3 * 25$  is  $-75$ . Then,  $-30 * (-5)$  is positive 150. And the last term is  $-81$ . So adding them up:  $-75 + 150 - 81$ . Let's calculate this.

First,  $-75 + 150 = 75$ . Then,  $75 - 81 = -6$ . So  $n = -6$ . Therefore, the vertex is at  $(-5, -6)$ , so  $n$  is  $-6$ .

884

Wait, let me double-check my calculations to make sure I didn't make any arithmetic errors. Let's redo the substitution.

Calculating  $y$  when  $x = -5$ :

First term:  $-3 * (-5)^2$ . The exponent first:  $(-5)^2 = 25$ . Then multiply by  $-3$ :  $-3 * 25 = -75$ . Second term:  $-30 * (-5)$ . Multiply those:  $-30$  times  $-5$  is  $150$ . Third term is  $-81$ . So adding them:  $-75 + 150 = 75$ ;  $75 - 81 = -6$ . Yep, that's correct.

Alternatively, maybe I can use another method to confirm. Sometimes, completing the square can be a good way to find the vertex. Let me try that. The given equation is  $y = -3x^2 - 30x - 81$ . To complete the square, first factor out the coefficient of  $x^2$  from the first two terms. So:

$y = -3(x^2 + 10x) - 81$ . Wait, let's check:  $-3x^2 - 30x$  becomes  $-3(x^2 + 10x)$ . Yes, that's correct because  $-3$  times  $x^2$  is  $-3x^2$  and  $-3$  times  $10x$  is  $-30x$ . Then, we have the remaining constant term as  $-81$ .

Now, to complete the square inside the parentheses, we need to take half of the coefficient of  $x$ , which is  $10$ , so half of  $10$  is  $5$ , and then square it:  $5^2 = 25$ . So we add and subtract  $25$  inside the parentheses to maintain equality. However, since we are adding  $25$  inside the parentheses that's multiplied by  $-3$ , we have to balance that by actually adding and subtracting  $-3 * 25$  to the entire equation. Wait, maybe I need to write it out more carefully.

So:

$$y = -3(x^2 + 10x + 25 - 25) - 81$$

Wait, that is equivalent to:

$$y = -3[(x^2 + 10x + 25) - 25] - 81$$

Which simplifies to:

$$y = -3(x + 5)^2 + 75 - 81$$

Because  $(x^2 + 10x + 25)$  is  $(x + 5)^2$ , and then we have  $-3$  times  $-25$  which is  $+75$ . Then, subtract  $81$ . So:

$$y = -3(x + 5)^2 + 75 - 81$$

Then, combining the constants:  $75 - 81 = -6$ . So the equation becomes:

$$y = -3(x + 5)^2 - 6$$

Which is in vertex form:  $y = a(x - h)^2 + k$ , where the vertex is  $(h, k)$ . Wait, in this case, it's  $(x + 5)$  which is the same as  $(x - (-5))$ , so  $h$  is  $-5$  and  $k$  is  $-6$ . Therefore, the vertex is at  $(-5, -6)$ , so  $n$  is  $-6$ . That confirms the previous result.

Alternatively, maybe I can use calculus, taking the derivative and setting it to zero, but since this is a simple quadratic, that might be overkill. But for thoroughness, let's try it. The derivative of  $y$  with respect to  $x$  is  $dy/dx = 2 * (-3)x + (-30) = -6x - 30$ . Setting this equal to zero for critical points:  $-6x - 30 = 0$ . Solving for  $x$ :  $-6x = 30 \Rightarrow x = -5$ . Then substitute  $x = -5$  into original equation, as before, gives  $y = -6$ . So again, same result.

Therefore, all three methods (vertex formula, completing the square, calculus) lead to the same answer:  $n = -6$ .

Is there any chance I made a mistake in the signs? Let me check again. The original equation is  $y = -3x^2 - 30x - 81$ . So  $a = -3$ ,  $b = -30$ . Then the  $x$ -coordinate of vertex is  $-b/(2a) = -(-30)/(2 * (-3)) = 30/-6 = -5$ . That's correct. Then substituting  $x = -5$  into the equation:  $y = -3 * (-5)^2 - 30 * (-5) - 81$ . Compute each term:

First term:  $-3 * (25) = -75$

Second term:  $-30 * (-5) = +150$

Third term:  $-81$

Total:  $-75 + 150 = 75$ ;  $75 - 81 = -6$ . Correct.

Completing the square also confirmed the same. Calculus as well. So multiple methods confirm that  $n = -6$ .

Therefore, I'm confident that the answer is  $n = -6$ .

886

887 Next, we demonstrate the output of step extraction. As explained in A.2, the output from step  
888 extraction contains line numbers corresponding to each plan and execution within a component. In  
889 the sample below, we replace line numbers with the corresponding block of text from the reasoning  
890 trajectory for ease of understanding. For the sake of brevity, we only show the full details for a few  
891 components.

#### Formatted output of Step Extraction

### Component 1

**Description:** Planning how to find the vertex of the parabola.

**Plan:** Okay, so I need to find the vertex ... Then once we have the  $x$ -coordinate, we can plug that back into the original equation to find the corresponding  $y$ -coordinate, which would be  $n$  in this case.

**Prompt:** Identify the values of  $a$ ,  $b$ , and  $c$  from the given quadratic equation.

**Execution:** So let's identify  $a$ ,  $b$ , and  $c$  ... it looks like  $a = -3$ ,  $b = -30$ , and  $c = -81$ .

**Comment:** No lines found

### Component 2

**Description:** Verifying the identified values of  $a$ ,  $b$ , and  $c$ .

...

### Component 3

**Description:** Calculating the  $x$ -coordinate of the vertex using the vertex formula.

...

### Component 4

**Description:** Calculating the  $y$ -coordinate of the vertex by substituting the  $x$ -coordinate.

...

892

```

893 ### Component 5
894 Description: Verifying the calculation of the  $y$ -coordinate.
895 ...
896
897 ### Component 6
898 Description: Using the method of completing the square to find the vertex.
899 Plan: Alternatively, maybe I can use another method to confirm. Sometimes, completing the
900 square can be a good way to find the vertex. Let me try that.
901 Prompt: Use the method of completing the square on the given equation to find the vertex.
902 Execution: The given equation is  $y = -3x^2 - 30x - 81$ . To complete the square, first factor
903 out the coefficient of  $x^2$  ... So the equation becomes:  $y = -3(x + 5)^2 - 6$  which is in vertex
904 form:  $y = a(x - h)^2 + k$ , where the vertex is  $(h, k)$  ... Therefore, the vertex is at  $(-5, -6)$ ,
905 so  $n$  is  $-6$ .
906 Comment: That confirms the previous result.
907 ...
908
909 ### Component 7
910 Description: Using calculus to find the vertex by taking the derivative and setting it to zero.
911 ...
912
913 ### Component 8
914 Description: Comparing results from different methods.
915 ...
916
917 ### Component 9
918 Description: Final verification of the solution and confirming results.
919 ...

```

In Figure 7, we demonstrate the dependencies that are inferred from running DAG creation over the steps (a.k.a components) extracted above. The model has identified that Components 2, 3, 6, and 7 can run in parallel. This is because once the values of  $a$ ,  $b$ , and  $c$  are identified in Component 1, the verification of that calculation (Component 2) and the computation of the  $x$ -coordinate (Component 3) are mutually independent. Besides, alternative approaches such as completing the square (Component 6) and using calculus (Component 7) are not dependent on components other than Component 1. Component 4 depends on Component 3 as the computation of the  $y$ -coordinate depends on the value of the  $x$ -coordinate identified in Component 3. Comparing the results of the three approaches (Component 8) depends on the results from Components 4, 6, and 7 while the final verification (Component 9) depends on Component 8. Hence, the DAG effectively models the dependencies among the components and identifies parallelization opportunities.

As explained in Appendix A.2, for components with short executions, we merge the executions with their plans. Hence, for the above problem, the executions of Components 2, 3, and 8 are merged with their plans. The trajectory containing plans and executions generated for the above problem following this modification is given below. Note that the plans are ordered based on the dependencies modeled in the DAG.

The sequential token count for the original response from R1 is measured as the total number of tokens in the response which is 1645. The trajectory sample below has a smaller sequential token count of 1445 due to the parallelized executions.

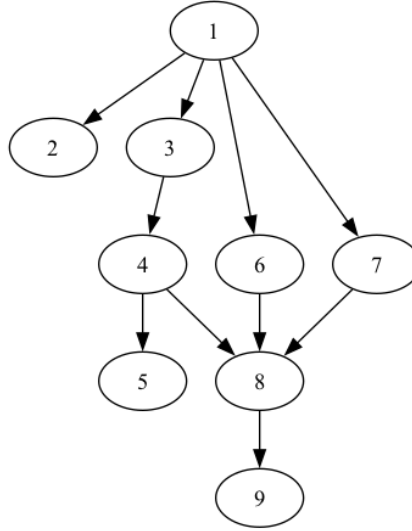


Figure 7: Dependencies inferred from running DAG creation over the steps extracted from the example reasoning trajectory in Appendix B.1

### Trajectory sample for fine-tuning

<Plan\_1>

Okay, so I need to find the vertex ... it looks like  $a = -3$ ,  $b = -30$ , and  $c = -81$ .

...

Wait, is that right? Let me check again ... The constant term is  $-81$ , so  $c = -81$ .

...

Now, using the formula and plugging in the values of  $a$  and  $b$ ... Let's compute that step by step.

...

<prompt\_1.1> **Calculate the x-coordinate of the vertex** </prompt\_1.1>

Alternatively, maybe I can use another method to confirm. Sometimes, completing the square can be a good way to find the vertex.

<prompt\_1.2> **Use the method of completing the square on the given equation to find the vertex.** </prompt\_1.2>

Alternatively, maybe I can use calculus, taking the derivative ... let's try it.

<prompt\_1.3> **Use calculus to find the x-coordinate** </prompt\_1.3>

</Plan\_1>

<Execution\_1>

```

<execution_1.1>
The numerator is  $-(-30)$ ... So  $m = 30/(-6) = -5$ . Therefore, the  $x$ -coordinate
of the vertex is  $-5$ .
</execution_1.1>

<execution_1.2>
The given equation is  $y = -3x^2 - 30x - 81$ . To complete the square, first factor
out the coefficient of  $x^2$  ... So the equation becomes:  $y = -3(x + 5)^2 - 6$  which is
in vertex form:  $y = a(x - h)^2 + k$ , where the vertex is  $(h, k)$  ... Therefore, the
vertex is at  $(-5, -6)$ , so  $n$  is  $-6$ .
</execution_1.2>

<execution_1.3>
The derivative of  $y$  with respect to  $x$  is  $dy/dx = 2 * (-3)x + (-30) = -6x - 30$ .
Setting this equal to zero for critical points ... as before, gives  $y = -6$ .
</execution_1.3>

</Execution_1>

<Plan_2>

Now, to find the  $y$ -coordinate  $n$  ... , I need to substitute  $x = -5$ ...
<prompt_2.1> Substitute  $x = -5$  to find the  $y$ -coordinate of the vertex.
</prompt_2.1>
</Plan_2>

<Execution_2>
<execution_2.1>
Plugging in  $x = -5$ :
...
So  $n = -6$ .
</execution_2.1>
</Execution_2>

<Plan_3>
Based on execution_2.1:
Wait, let me double-check my calculations ...
<prompt_3.1> Redo the substitution of  $x = -5$  into the original equation to
verify. </prompt_3.1>

Based on execution_2.1, execution_1.2, execution_1.3:
Therefore, all three methods (vertex formula, completing the square, calculus) lead
to the same answer:  $n = -6$ .

Let me check again.
<prompt_3.2> Recheck the calculations and confirm the results
</prompt_3.2>

</Plan_3>

<Execution_3>

```

<execution\_3.1>

Calculating  $y$  when  $x = -5$ : First term:  $-3 * (-5)^2$  ... So adding them:  
 $n = -75 + 150 = 75$ ;  $75 - 81 = -6$ .

</execution\_3.1>

<execution\_3.2>

The original equation is ... So multiple methods confirm that  $n = -6$ .

</execution\_3.2>

</Execution\_3>

<Final\_answer>

Therefore, the value of  $n$  is  $\boxed{-6}$ .

</Final\_answer>

915

916 **B.2 Demonstration of SPRINT execution versus sequential reasoning from RFT**

917 Given below is an example of how SPRINT responds to a question from the MATH500 benchmark at  
 918 inference time. The executions presented side-by-side are carried out in parallel using the inference  
 919 strategy shown in Figure 3.

920 **Question:** The binary number  $10101001110_2$  is equal to what number in base eight?

**Planning and parallel execution trajectory generated by SPRINT**

<Plan\_1>

Okay, so I need to convert the binary number  $10101001110_2$  to base eight. So, the plan is probably to group the binary digits into sets of three starting from the right and then convert each group to an octal digit.

Wait, but first, let me confirm: the binary number given is  $10101001110$ . Let me count the digits.

Let's see:  $10101001110$ . That's 11 binary digits. Since 8 is  $2^3$ , each octal digit is three bits, so 11 bits would need to be grouped into 4 octal digits (since  $4 \times 3 = 12$ , which is more than 11) but we need to add a leading zero to make it 12 bits? Wait, no, actually, when converting from binary to octal, you start from the right and split into groups of three. Right, so even if the total number of bits isn't a multiple of three, you can add leading zeros to make it a multiple of three. Let me check.

<prompt\_1.1> **Verify the method of converting binary to octal by considering the binary number  $10101001110$ .** </prompt\_1.1>

But let me think again. Let me confirm with another approach. Alternatively, I could convert the binary number to decimal first and then to octal. Maybe that's a good way to cross-verify.

<prompt\_1.2> **Consider converting the binary number to decimal first and then to octal as an alternative method.** </prompt\_1.2>

</Plan\_1>

<Execution\_1>

<execution\_1.1>

The original number is  $10101001110$ . Let's write them out: positions from right to left (since we start grouping from the right). Wait, actually, when grouping, we start from the right, so the leftmost digit might need to be padded with zeros to make the total number of digits a multiple of three. So the binary number has 11 digits. 11 divided by 3 is 3 with a remainder of 2. So, we need to add one leading zero to make it 12 digits. Wait,  $3 \times 4 = 12$ , so yes, adding one zero on the left. So the original binary is  $10101001110$ . Adding a leading zero gives  $010101001110$ .

</execution\_1.1>

<execution\_1.2>

Let's see. The binary number is  $10101001110$ . Let me write down the positions from right to left, starting at 0. Wait, actually, in computers, the rightmost bit is the least significant bit, which is position 0. So let's list the bits: Position: 10 9 8 7 6 5 4 3 2 1 0 Bit: 1 0 1 0 1 0 0 1 1 1 0 ...  
 So: Bit 10:  $1 (2^{10})$   
 Bit 9:  $0 (2^9)$   
 Bit 8:  $1 (2^8)$   
 ...

So the binary number is:  $2^{10} + 2^8 + \dots 2^1$   
 ... Adding them all up:  $1024 + 256 = 1280$ ;  $1280 + 64 = 1344$ ;  $1344 + 8 = 1352$ ;  $1352 + 4 = 1356$ ;  $1356 + 2 = 1358$ . So the decimal equivalent is 1358. </execution\_1.2>

</Execution\_1>

<Plan\_2>

Based on execution\_1.1:

Now, splitting into groups of three starting from the right:

<prompt\_2.1> **Split the 12-digit binary number into groups of three starting from the right and convert each group to octal.** </prompt\_2.1>

Based on execution\_1.2:

Now, converting 1358 to octal. Let's divide by 8:

<prompt\_2.2> **Convert the decimal number 1358 to octal using division by 8.** </prompt\_2.2>

</Plan\_2>

<Execution\_2>

<execution\_2.1>

0 1 0 1 0 0 0 1 1 1 0

Wait, no, wait. Wait, the binary number after adding a leading zero is 010101001110. Now, group into 4 sets of three: Group 1 (leftmost): 010, Group 2: 101, Group 3: 001, Group 4: 110. Then convert each of these to octal digits. Wait, octal digits go from 0 to 7, so each group of three bits represents 0-7. Let me recall the binary to octal mapping:

000 = 0, 001 = 1, 010 = 2, ... So:

Group 1: 010 → 2

Group 2: 101 → 5

Group 3: 001 → 1

Group 4: 110 → 6

So putting them together: 2 5 1 6 → 2516<sub>8</sub>.

</execution\_2.1>

<execution\_2.2>

1358 ÷ 8 = 169 with remainder 6  
(since 169\*8 = 1352, 1358 - 1352 = 6)

169 ÷ 8 = 21 with remainder 1  
(21\*8 = 168, 169 - 168 = 1)

21 ÷ 8 = 2 with remainder 5 (2\*8 = 16, 21 - 16 = 5)

2 ÷ 8 = 0 with remainder 2

So the remainders, from last to first, are 2, 5, 1, 6. Therefore, 1358 in octal is 2516<sub>8</sub>.  
</execution\_2.2>

</Execution\_2>

<Plan\_3>

Based on execution\_2.1:

Wait, but let me check again. Let's verify the grouping.

<prompt\_3.1> **Verify the grouping of the binary digits and the conversion to octal.** </prompt\_3.1>

Based on execution\_2.2:

Therefore, the binary number 10101001110<sub>2</sub> is equal to 2516<sub>8</sub>. So both methods give the same result, which is reassuring.

<prompt\_3.2> **Reflect on the solution and confirm its correctness by comparing both conversion methods.** </prompt\_3.2>

</Plan\_3>

<Execution\_3>

```

<execution_3.1>
Original binary with leading
zero: 010 101 001 110.
Group 1: 010 → 2
Group 2: 101 → 5
Group 3: 001 → 1
Group 4: 110 → 6
So octal number is 2 5 1 6
→ 2516.
</execution_3.1>

```

```

<execution_3.2>
I think this confirms that the answer is correct.
But let me just go through the steps once more
to be thorough. Original binary: 10101001110
Convert to groups of three starting from the right.
Since it's 11 digits, add a leading zero to make
it 12 digits: 010101001110.
Group into four parts: 010 101 001 110. Convert
each:
010 = 2, 101 = 5, 001 = 1, 110 = 6
Combine: 2516. Yep.
Alternatively, converting to decimal via binary
to decimal conversion:  $10101001110_2 = 1 * 2^{10} + 0 * 2^9 + 1 * 2^8 + 0 * 2^7 + \dots + 0 * 2^0 = 1024 + 256 + 64 + 8 + 4 + 2 = 1358$ . Then
1358 divided by 8 gives  $169 * 8 = 1352$ , remainder
6; 169 divided by 8 is  $21 * 8 = 168$ ,
... So 2 5 1 6. So 2516. All checks out. There-
fore, the final answer is 2516 in octal. I think
that's solid. No mistakes detected in the process.
</execution_3.2>

```

```

</Execution_3>

```

```

<Final_answer>

```

This step-by-step conversion shows that the binary number  $10101001110_2$  is equal to  $\boxed{2516_8}$  in base eight.

```

</Final_answer>

```

923

924 The sequential token count of the above response from SPRINT is 1463. For the same question, we  
 925 present the sequential trajectory generated by the RFT model below. The sequential token count  
 926 of this response is 1989 since all tokens are generated sequentially. This highlights the benefit of  
 927 parallelizing executions using SPRINT.

#### Sequential reasoning trajectory generated by the RFT model

Okay, so I need to convert the binary number  $10101001110_2$  to base eight. Hmm, I remember that converting between binary and octal is done by grouping the binary digits into sets of three, starting from the right. Each group of three bits corresponds to a single octal digit. Let me think through the steps carefully.

First, let me write down the binary number: 1 0 1 0 1 0 0 1 1 1 0. Wait, let me count the digits. The original number is 10101001110, which has 11 digits. Since octal groups are in threes, I need to split this into groups of three. But 11 isn't divisible by three. So, I should add leading zeros to make the total number of digits a multiple of three. Right, so 11 divided by 3 is 3 with a remainder of 2. That means I need to add one leading zero to make it 12 digits. So the new number becomes  $010101001110_2$ .

Now, let me split this into groups of three starting from the right. Let me write it out:

Group 1: 010, Group 2: 101, Group 3: 001, Group 4: 110

Wait, no. Wait, the leading zeros are added at the beginning, so the original number was 10101001110. Adding a leading zero makes it 010101001110. Now, splitting into groups of

928

three from the right:

Starting from the end, the last three digits are 110, then moving left, the next three are 001, then 101, then 010. So the groups are 010, 101, 001, 110. Now, each of these binary groups needs to be converted to their octal equivalents.

Let me convert each group:

$010_2$  is 2 in octal because  $0 \times 4 + 1 \times 2 + 0 \times 1 = 2$ .

$101_2$  is 5 in octal because  $1 \times 4 + 0 \times 2 + 1 \times 1 = 5$ .

...

So putting those together in order from left to right, the octal digits are 2, 5, 1, 6. Therefore, the octal number should be  $2516_8$ . Wait, but let me verify this again to make sure I didn't make a mistake in grouping or conversion.

Wait, when grouping from the right, the first group (rightmost) is 110, then moving left, the next is 001, then 101, then 010. So the groups are 010, 101, 001, 110. Converting each to octal:

$010_2 = 2$ ,  $101_2 = 5$ ,  $001_2 = 1$ ,  $110_2 = 6$ . So the octal number is 2 5 1 6, which is  $2516_8$ . Let me check if this is correct by converting the original binary number to decimal and then converting that decimal number to octal to see if it matches.

First, converting  $10101001110_2$  to decimal. Let's write out the binary number with each digit's place value:

From right to left, the positions are 0 to 10. The digits are:

Position 10: 1, Position 9: 0, Position 8: 1, ..., Position 0: 0

Calculating the decimal value:

$$1 \times (2^{10}) + 0 \times (2^9) + 1 \times (2^8) + 0 \times (2^7) + 1 \times (2^6) + 0 \times (2^5) + 0 \times (2^4) + 1 \times (2^3) + 1 \times (2^2) + 1 \times (2^1) + 0 \times (2^0)$$

Calculating each term:

$$2^{10} = 1024, 2^8 = 256, 2^6 = 64, 2^3 = 8, 2^2 = 4, 2^1 = 2$$

Adding them up:  $1024 + 256 = 1280$ ;  $1280 + 64 = 1344$ ;  $1344 + 8 = 1352$ ;  $1352 + 4 = 1356$ ;  $1356 + 2 = 1358$ .

So the decimal value is 1358. Now, converting 1358 to octal. Let's divide 1358 by 8 repeatedly.

$$1358 \div 8 = 169 \text{ with a remainder of } 6.$$

$$169 \div 8 = 21 \text{ with a remainder of } 1.$$

$$21 \div 8 = 2 \text{ with a remainder of } 5.$$

$$2 \div 8 = 0 \text{ with a remainder of } 2.$$

Writing the remainders from last to first: 2 5 1 6. So  $2516_8$ . Which matches the previous result. Therefore, the conversion is correct.

But wait, let me check the grouping again. The original binary number was 10101001110. When adding a leading zero, it becomes 010101001110. Splitting into groups of three from

the right:

Starting from the right:

111 001 101 010. Wait, that's different. Wait, perhaps I made a mistake in grouping.

Wait, the original binary number is 10101001110. Let me write it as 10 101 001 110. Wait, that's not correct. Wait, the original number is 11 digits. So when adding a leading zero, it becomes 12 digits: 0 1 0 1 0 1 0 0 1 1 1 0. Then grouping from the right:

Group1: 0 1 0 (positions 11, 10, 9), Group2: 1 0 1 (positions 8, 7, 6), Group3: 0 0 1 (positions 5, 4, 3), Group4: 1 1 0 (positions 2, 1, 0)

Wait, but when grouping from the right, the rightmost group is the last three digits. So the original number is 010101001110. The rightmost three digits are 110, then moving left, next three are 001, then 101, then 010. So the groups are 010, 101, 001, 110. Which is the same as before. So converting each gives 2, 5, 1, 6. So  $2516_8$ . Which is correct. So the answer is  $2516_8$ .

But wait, when I grouped the original number as 10101001110, which is 11 digits, adding a leading zero makes it 010101001110, which is 12 digits. So grouping into three from the right: 010, 101, 001, 110. So that's correct.

Alternatively, another way to check is by converting each group as I did before. So yes, the answer is 2516 in octal.

But let me just confirm once again by converting  $2516_8$  back to binary to see if it's  $10101001110_2$ .

2516 in octal:

Each digit corresponds to three binary digits:

2  $\rightarrow$  010

5  $\rightarrow$  101

1  $\rightarrow$  001

6  $\rightarrow$  110

So putting them together: 010 101 001 110  $\rightarrow$  010101001110, which is the original number. So that's correct.

Therefore, the final answer is  $\boxed{2516}$  in base eight.