

# When Is Thinking Enough? Early Exit via Sufficiency Assessment for Efficient Reasoning

Anonymous ACL submission

## Abstract

Large reasoning models (LRMs) have achieved remarkable performance in complex reasoning tasks, driven by their powerful inference-time scaling capability. However, LRMs often suffer from overthinking, which results in substantial computational redundancy and significantly reduces efficiency. Early-exit methods aim to mitigate this issue by terminating reasoning once sufficient evidence has been generated, yet existing approaches mostly rely on handcrafted or empirical indicators that are unreliable and impractical. In this work, we introduce **Dynamic Thought Sufficiency in Reasoning (DTSR)**, a novel framework for efficient reasoning that enables the model to dynamically assess the sufficiency of its chain-of-thought (CoT) and determine the optimal point for early exit. Inspired by human metacognition, DTSR operates in two stages: (1) *Reflection Signal Monitoring*, which identifies reflection signals as potential cues for early exit, and (2) *Thought Sufficiency Check*, which evaluates whether the current CoT is sufficient to derive the final answer. Experimental results on the Qwen3 models show that DTSR reduces reasoning length by 28.9%–34.9% with minimal performance loss, effectively mitigating overthinking. We further discuss overconfidence in LRMs and self-evaluation paradigms, providing valuable insights for early-exit reasoning.

## 1 Introduction

The success of large reasoning models (LRMs), such as OpenAI’s o1/o4 (OpenAI, 2024, 2025), Gemini 2.5 Pro (Comanici et al., 2025), DeepSeek-R1 (Guo et al., 2025), and Qwen3 (Yang et al., 2025a), represents a significant breakthrough in the evolution of LLMs. LRMs engage in long chain-of-thought (CoT) reasoning before producing answers. The inference-time scaling capability (Chen et al., 2025a; Ji et al., 2025) allows them to achieve remarkable performance in challenging reasoning

tasks (e.g. mathematics, programming, and scientific question answering), and show human-like reasoning patterns, such as problem decomposition, verification and backtracking.

Nevertheless, recent studies (Chen et al., 2025b; Cuadron et al., 2025) have shown that LRMs often suffer from overthinking: even on simple problems, they tend to consume excessive tokens by repeatedly verifying answers or exploring alternative strategies after reaching the correct solution. Such redundant reasoning is unnecessary and leads to inefficiency. To address this, several prior works have proposed early-exit strategies that stop the reasoning process once the LRM has generated sufficient evidence to solve the problem. The main challenge of early-exit strategies is to determine the optimal exit point, where LRMs achieve near-lossless performance with minimal reasoning cost. Dynasor-CoT (Fu et al., 2025) first employs the consistency of intermediate answers to determine when to exit reasoning; however, it still generates additional token intervals after producing the correct answer to verify its correctness, thereby missing the optimal exit point. DEER (Yang et al., 2025b) and CGRS (Huang et al., 2025) employ the entropy of the intermediate result as a confidence measure to determine whether LRMs reach the optimal exit point. Nevertheless, using answer confidence as a criterion for reasoning exit is not always reliable, as reasoning models can exhibit overconfidence even when their answers are incorrect (Mei et al., 2025). Moreover, such methods that rely on probing intermediate answers are limited to tasks with short, fixed-format answers. For tasks involving long answers or open-ended generation without a definitive answer, intermediate answers are difficult to extract, rendering these methods impractical.

To overcome these limitations, this work aims to establish a reliable and generalizable criterion for early-exit. Unlike LLM reasoning, human cognition involves metacognitive (Flavell, 1979; Fleur

084 et al., 2021) abilities, which allow people to mon- 134  
085 itor, evaluate, and regulate their thoughts, and to 135  
086 stop thinking once the process is deemed sufficient. 136  
087 Inspired by this, we wonder whether LRMs can 137  
088 similarly determine the optimal point to exit rea- 138  
089 soning by continuously assessing the sufficiency 139  
090 of their current thoughts? To answer this, we pro- 140  
091 pose Dynamic Thought Sufficiency in Reasoning 141  
092 (DTSR), a novel framework that enables models 142  
093 to dynamically evaluate whether their current CoT 143  
094 is sufficient to solve the problem. DTSR consists 144  
095 of two stages: 1) Reflection Signal Monitoring: 145  
096 LRMs often exhibit reflective behaviors after pro- 146  
097 ducing an answer, typically triggered by specific 147  
098 reflection signals. We identify these signals as po- 148  
099 tential exit points for reasoning. 2) Thought Suffi-  
100 ciency Check: once a reflection signal is detected,  
101 the model evaluates the sufficiency of its current  
102 reasoning. If the sufficiency score is high, reason-  
103 ing stops and the solution is output; otherwise, the  
104 model continues along the reasoning path until the  
105 next reflection signal occurs.

106 To evaluate the effectiveness of our proposed  
107 method, we conducted extensive experiments  
108 across multiple LRMs. The results show that DTSR  
109 significantly outperforms baseline approaches,  
110 achieving a favorable balance between accuracy  
111 and efficiency. This also demonstrates that rea-  
112 soning models can reliably assess the sufficiency  
113 of their thought, providing valuable insights and  
114 guidance for advancing efficient reasoning.

115 Overall, our contributions are as follows:

- 116 • We propose a novel framework for efficient rea- 167  
117 soning that enables the model to dynamically 168  
118 assess the sufficiency of its CoT and determine 169  
119 the optimal point for early exit. 170
- 120 • Extensive experiments conducted on Qwen3 se- 171  
121 ries models of various sizes demonstrate that 172  
122 DTSR consistently outperforms baseline meth- 173  
123 ods, achieving a 28.9%–34.9% reduction in gen- 174  
124 eration length with almost no loss in performance, 175  
125 thereby significantly mitigating overthinking. 176
- 126 • Further discussion of overconfidence in LRMs 177  
127 and the self-evaluation paradigm offers a fresh 178  
128 perspective and valuable insights for guiding 179  
129 early-exit reasoning. 180

## 130 2 Methodology

### 131 2.1 Preliminaries

132 **Generation Patterns of Reasoning Models** The  
133 responses of LRMs are typically structured in two

134 stages: a reasoning phase, in which the model en- 135  
136 gages in slow, human-like thinking, encapsulated 137  
138 within the <think> and </think> delimiters, fol- 139  
140 lowed by a solution phase that summarizes the 141  
142 reasoning and produces the final answer. During 143  
144 the thinking process, the model exhibits complex 145  
146 reasoning behaviors, such as problem decomposi- 147  
148 tion, strategy exploration, output verification, and 149  
150 self-correction (Gandhi et al., 2025; Bogdan et al., 151  
152 2025). To trigger these reflective behaviors, LRMs 153  
154 often generate tokens like “Wait”, “Alternatively”, 155  
156 or “Let me check” as cues. However, once the 157  
158 correct answer has been reached, further verifica- 159  
160 tion becomes redundant, leading to unnecessary 160  
161 reasoning cost (Chen et al., 2025b). 162

163 **Early Exit in Reasoning** Early exit is a promis- 164  
165 ing strategy for mitigating redundant reasoning. 166  
167 Muennighoff et al. (2025) first introduces a simple 168  
169 budget forcing method: when generation reaches 170  
171 the maximum token budget, the model is forced to 172  
173 append the end-of-thinking token and “Final An- 174  
175 swer:” to terminate reasoning and output its current 175  
176 answer. Dynasor-CoT (Fu et al., 2025) probes in- 176  
177 termediate answers via designed prompts at fixed 177  
178 intervals and terminates reasoning once consecu- 178  
179 tive answers are consistent. It partially mitigates 179  
180 overthinking but still generates extra token inter- 181  
182 vals to verify correctness after the correct answer 182  
183 is produced, failing to achieve optimal exit point. 183  
184 In contrast, DEER (Yang et al., 2025b) outputs in- 184  
185 termediate answers at reasoning transition points 185  
186 (e.g., “Wait”) and uses entropy to estimate their con- 186  
187 fidence, terminating the reasoning process when 187  
188 achieving high confidence. 188

189 However, previous methods mainly rely on hand- 189  
190 crafted or statistical criteria, which are unreliable 190  
191 for determining whether the current reasoning pro- 191  
192 cess is sufficient to yield the correct answer. Mei 192  
193 et al. (2025) find that even state-of-the-art reason- 193  
194 ing models are overconfident, particularly for incor- 194  
195 rect answers, and this overconfidence grows with 195  
196 the longer reasoning process. Consequently, a sub- 196  
197 stantial gap exists between token probabilities and 197  
198 the actual correctness of answers. From a cognitive 198  
199 perspective, humans rarely produce explicit inter- 199  
200 mediate answers to decide when to stop thinking. 200  
201 Instead, through metacognitive processes (Fleur 201  
202 et al., 2021), they internally assess whether their 202  
203 thoughts are sufficient to reach a final answer. In- 203  
204 spired by this, enabling reasoning models to self- 204  
205 assess the sufficiency of their thought to determine 205

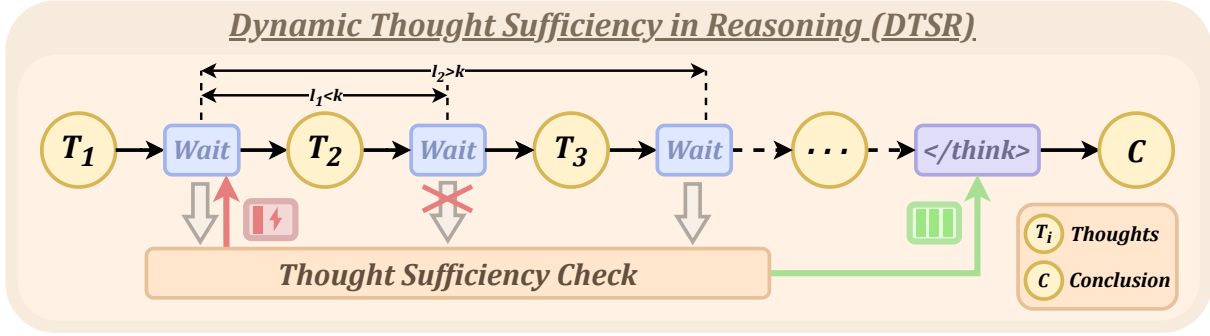


Figure 1: Overview of the Dynamic Thought Sufficiency in Reasoning (DTSR) method. Upon a reflection signal, the model evaluates the sufficiency of its previous thought. If the thought is insufficient, reasoning continues; checks are skipped within  $k$  tokens. Once the thought is sufficient, `</think>` is appended and a conclusion is generated.

185 if it can yield a final answer is highly intuitive. 220  
 186 Therefore, we propose Dynamic Thought Sufficiency 221  
 187 in Reasoning (DTSR), a method that enables 222  
 188 models to assess whether their current thought is 223  
 189 sufficient to derive the final answer. 224

## 190 2.2 Our Proposal: Dynamic Thought 226 191 Sufficiency in Reasoning 227

192 As shown in Figure 1, we present DTSR, a method 228  
 193 that enables the model to dynamically evaluate the 229  
 194 sufficiency of its reasoning process. Our method 230  
 195 consists of two stages: reflection signal monitoring 231  
 196 and thought sufficiency check. 232

197 **Reflection Signal Monitoring** Our objective is 234  
 198 to exit the reasoning process at the optimal point. 235  
 199 Since assessing thought sufficiency after every gener- 236  
 200 ated token is computationally impractical, we 237  
 201 instead seek to identify potential exit signals that 238  
 202 indicate when verification should be triggered. To 239  
 203 this end, we conduct an empirical analysis of reason- 240  
 204 ing trajectories generated by Qwen3-32B. For 241  
 205 each trajectory, we identify its optimal early-exit 242  
 206 point, defined as the earliest position in the reason- 243  
 207 ing process at which the model is already capable 244  
 208 of producing a correct answer (Figure 7). We find 245  
 209 that these optimal exit points are often followed 246  
 210 by explicit self-reflection or answer-verification be-  
 211 haviors, manifested through cues such as “Wait”,  
 212 “Alternatively”, and phrases beginning with “But”  
 213 (e.g., “But wait”, “But let me check”). Based on  
 214 these observations, we construct a reflection signal  
 215 set, the complete list of which is shown in Table 4.

216 **Thought Sufficiency Check** The LRM continues 253  
 217 reasoning until a reflection signal is encountered. 254  
 218 Upon detecting such a signal, the model performs 255  
 219 a thought sufficiency check on the reasoning trace 256

220 generated so far to determine whether it can ter-  
 221minate further thinking and produce a final solu-  
 222tion. Specifically, at each checkpoint, the original  
 223question together with the current chain-of-thought  
 224(CoT) is fed into a dedicated thought sufficiency  
 225check prompt template (shown in Figure 8). The  
 226model is then prompted to generate a scalar score  
 227indicating the sufficiency of the current reasoning.  
 228A score exceeding the threshold  $\tau$  indicates that the  
 229model deems its reasoning sufficient; in this case,  
 230we append a termination marker `</think>` to halt  
 231further reasoning and allow the model to output the  
 232final answer. Conversely, if the score falls below  
 233the threshold  $\tau$ , the reasoning is considered insuffi-  
 234cient, and the model resumes generation from the  
 235last reflection signal, continuing the reasoning pro-  
 236cess until the next checkpoint. This procedure is  
 237repeated until the reasoning process is complete.  
 238Unlike prior approaches that rely on handcrafted  
 239heuristics or empirical external metrics, our method  
 240determines the reasoning exit point through the  
 241model’s internal self-evaluation. In this paradigm,  
 242the model assesses the sufficiency of its previously  
 243generated reasoning from a third-person perspec-  
 244tive, treating the reasoning trace as an external ob-  
 245ject rather than directly evaluating itself. Further  
 246discussion of this can be found in Section 4.4.

247 Moreover, reflection signals frequently occur 247  
 248 in close succession, causing repeated sufficiency 248  
 249 checks to yield similar outcomes and resulting in 249  
 250 redundant computation. To mitigate this issue, we 250  
 251 enforce a minimum token interval  $k$  between con- 251  
 252secutive sufficiency checks. Concretely, a reflec- 252  
 253tion signal triggers a thought sufficiency check only 253  
 254 if at least  $k$  tokens have been generated since the 254  
 255 previous check; otherwise, the current signal is 255  
 256 skipped and the process continues (Algorithm 1). 256

---

**Algorithm 1** Dynamic Thought Sufficiency in Reasoning (DTSR)

---

```
1: Init: model  $\mathcal{M}$ , question  $Q$ , prompt  $q_{cot}$ ,  
   reflection signals  $\mathcal{S}$ , min interval  $k$ , threshold  
    $\tau$ , max length  $max\_len$   
2:  $L \leftarrow q_{cot} + Q$ , reasoning trace  $T \leftarrow \emptyset$ ,  
   counter  $c \leftarrow 0$   
3: while  $len(L) < max\_len$  do  
4:   Generate token  $t$ ,  $T \leftarrow T + t$ ,  $c \leftarrow c + 1$   
5:   if  $t \in \mathcal{S}$  and  $c \geq k$  then  
6:     Build sufficiency prompt with  $(Q, T)$   
7:     Obtain score  $s \in [0, 100]$  from  $\mathcal{M}$   
8:     if  $s \geq \tau$  then  
9:        $T \leftarrow T + \langle /think \rangle$   
10:    else  
11:      Continue reasoning,  $c \leftarrow 0$   
12:    end if  
13:  end if  
14: end while  
15: return  $T$ 
```

---

### 3 Experiments

#### 3.1 Experimental Setup

**Benchmarks and Metrics** To comprehensively evaluate the reasoning capabilities of LRMs, we select six widely used benchmarks: GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), AMC 2023, OlympiadBench (He et al., 2024), GPQA Diamond (Rein et al., 2024) and LiveCodeBench (Jain et al., 2025). Our goal is to significantly reduce the number of tokens generated during the reasoning process while maintaining high reasoning performance. To this end, we employ two primary metrics: Accuracy (Acc) and Token Count (Tok). Acc measures the proportion of correct final answers, calculated as the average pass@1 score, whereas Tok represents the average number of tokens generated per sample, serving as an indicator of reasoning efficiency.

**Models** We conduct extensive experiments on the Qwen3 (Yang et al., 2025a) series of models. Our selection spans different model sizes (8B, 14B, 32B) of reasoning models to validate the generalizability of our method. Despite their strong reasoning ability, these models often exhibit overthinking during the reasoning process.

**Baselines** We compare DTSR with the vanilla LRM as well as existing state-of-the-art training-free efficient reasoning methods, including No-

Thinking (Ma et al., 2025a), NOWAIT (Wang et al., 2025), and DEER (Yang et al., 2025b). Vanilla allows the model to generate answers directly without any intervention. NoThinking prompts the model to skip the reasoning stage and answer the question directly. NOWAIT suppresses explicit self-reflection by masking reflective tokens such as “wait” and “hmm” during the reasoning process. DEER makes early-exit decisions based on the confidence scores of intermediate answers. In addition, we compare DTSR with recent training-based methods (Arora and Zanette, 2025a; Dai et al., 2025).

**Implementation Details** All experiments are conducted using the vLLM framework to ensure high throughput and memory-efficient inference. For the decoding strategy, we set the temperature to 0.6 and top- $p$  to 0.95. The maximum generation length is set to 16k tokens, and each dataset is generated three times, with the final results reported as the average pass@1. In our method, the thought sufficiency threshold  $\tau$  is set to 100, and the minimum token interval  $k$  is set to 64.

#### 3.2 Experimental Results

**Overall Performance** The results in Table 1 show that DTSR consistently outperforms the baseline methods across three model sizes and five benchmark datasets. Compared with vanilla reasoning models, DTSR achieves almost the same level of accuracy without any degradation, while reducing sequence length by 28.9%–34.9%, leading to more efficient inference. Notably, when applied to Qwen3-14B, DTSR even improves performance on challenging reasoning tasks such as GPQA and OlympiadBench, underscoring its effectiveness.

**Comparison with SOTAs** We compare DTSR with several recent training-free efficient reasoning methods. As shown in Table 1, DTSR effectively shortens reasoning length without compromising accuracy, whereas other baselines struggle to balance efficiency with performance. For instance, NoThinking enforces models to bypass the reasoning process through prompting, which indeed shortens sequences but deviates from the natural generation paradigm and severely limits reasoning capability. NoWAIT mitigates redundant output by masking frequent reflective tokens, yet this restriction on the model’s inherent self-reflection results in significant performance degradation on complex reasoning tasks. DEER relies on the confidence scores of intermediate answers, yet models often exhibit

| Method             | GSM8K          |                  | MATH-500       |                  | AMC            |                  | GPQA           |                  | Olympiad       |                  | Overall                     |                               |
|--------------------|----------------|------------------|----------------|------------------|----------------|------------------|----------------|------------------|----------------|------------------|-----------------------------|-------------------------------|
|                    | Acc $\uparrow$ | Tok $\downarrow$ | Acc $\uparrow$ | Tok $\downarrow$ | Acc $\uparrow$ | Tok $\downarrow$ | Acc $\uparrow$ | Tok $\downarrow$ | Acc $\uparrow$ | Tok $\downarrow$ | Acc $\uparrow$              | Tok $\downarrow$              |
| <b>Qwen3-8B</b>    |                |                  |                |                  |                |                  |                |                  |                |                  |                             |                               |
| <i>Vanilla</i>     | 96.0           | 2208             | 94.4           | 5067             | 92.5           | 7232             | 60.1           | 8861             | 66.4           | 9182             | 81.9                        | 6510                          |
| <i>NoThinking</i>  | 92.6           | 303              | 86.0           | 1067             | 72.5           | 2264             | 53.0           | 1271             | 48.3           | 2385             | 70.5                        | 1458                          |
| <i>NoWAIT</i>      | 94.9           | 1307             | 92.8           | 3485             | 90.0           | 5448             | 55.6           | 6920             | 62.4           | 6998             | 79.1                        | 4832                          |
| <i>DEER</i>        | 95.1           | 963              | 93.4           | 2549             | 87.5           | 5155             | 57.1           | 8306             | 63.4           | 5768             | 79.3                        | 4532                          |
| <i>DTSR (ours)</i> | 95.1           | 1152             | 94.0           | 2359             | 90.0           | 4507             | 59.6           | 7942             | 66.4           | 6180             | <b>81.0</b> <sub>-0.9</sub> | <b>4428</b> <sub>-32.0%</sub> |
| <b>Qwen3-14B</b>   |                |                  |                |                  |                |                  |                |                  |                |                  |                             |                               |
| <i>Vanilla</i>     | 96.2           | 1672             | 95.4           | 4503             | 97.5           | 6562             | 65.7           | 7513             | 67.0           | 8555             | 84.4                        | 5761                          |
| <i>NoThinking</i>  | 95.1           | 278              | 88.2           | 852              | 77.5           | 2065             | 55.6           | 1286             | 51.7           | 2029             | 73.6                        | 1302                          |
| <i>NoWAIT</i>      | 95.8           | 1133             | 92.8           | 3307             | 97.5           | 5035             | 64.1           | 6775             | 65.6           | 6670             | 83.2                        | 4584                          |
| <i>DEER</i>        | 95.3           | 896              | 94.4           | 2601             | 97.5           | 4514             | 62.6           | 7105             | 65.1           | 6717             | 83.0                        | 4367                          |
| <i>DTSR (ours)</i> | 96.2           | 849              | 95.0           | 2247             | 97.5           | 3975             | 66.2           | 5916             | 68.9           | 5755             | <b>84.8</b> <sub>+0.4</sub> | <b>3748</b> <sub>-34.9%</sub> |
| <b>Qwen3-32B</b>   |                |                  |                |                  |                |                  |                |                  |                |                  |                             |                               |
| <i>Vanilla</i>     | 96.6           | 1707             | 94.8           | 4489             | 95.0           | 6807             | 68.2           | 6906             | 69.0           | 8280             | 84.7                        | 5638                          |
| <i>NoThinking</i>  | 94.8           | 283              | 86.4           | 958              | 70.0           | 1738             | 54.6           | 561              | 52.4           | 2076             | 71.6                        | 1123                          |
| <i>NoWAIT</i>      | 95.8           | 1212             | 93.4           | 3285             | 95.0           | 4894             | 66.7           | 5436             | 65.8           | 6598             | 83.3                        | 4285                          |
| <i>DEER</i>        | 95.5           | 979              | 94.0           | 2881             | 92.5           | 4966             | 67.2           | 6395             | 66.4           | 6370             | 83.1                        | 4318                          |
| <i>DTSR (ours)</i> | 96.0           | 849              | 95.0           | 2660             | 95.0           | 4617             | 69.2           | 5677             | 67.7           | 6249             | <b>84.6</b> <sub>-0.1</sub> | <b>4010</b> <sub>-28.9%</sub> |

Table 1: Experimental results across various reasoning models. "Acc" denotes accuracy, and "Tok" denotes token count.  $\uparrow$  indicates that higher values are better, while  $\downarrow$  indicates that lower values are better.

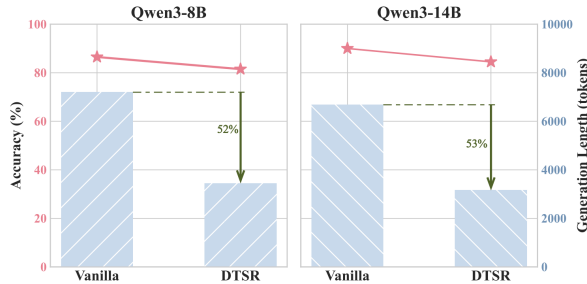


Figure 2: Performance of DTSR on LiveCodeBench.

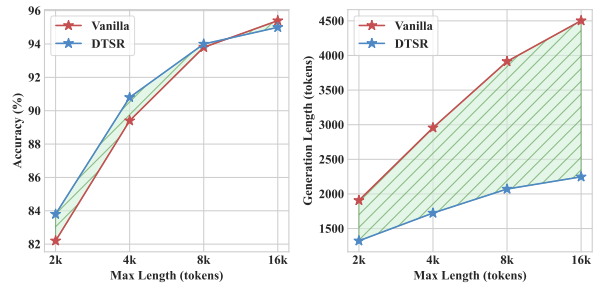


Figure 3: Comparison of Vanilla and DTSR on MATH-500 as the token budget increases.

high confidence even in incorrect ones. Moreover, it focuses only on partial answers while neglecting the global reasoning process, making it unreliable. In contrast, DTSR performs self-assessment over the entire reasoning process, which accounts for global coherence while mitigating overconfidence, thereby achieving superior overall performance.

We also present a comparison between DTSR and training-based methods in Table 5. The results show that DTSR achieves a generation length comparable to training-based approaches. Although DTSR exhibits slightly lower accuracy due to the absence of additional training, these results demonstrate that DTSR remains competitive with training-based methods while requiring no extra training.

**Performance on Programming Tasks** To evaluate the generalization of DTSR across different domains, we additionally assess its performance on programming tasks. Figure 2 presents the evalua-

tion results of Qwen-8B and Qwen-14B on LiveCodeBench. Compared to mathematical and scientific tasks, the reasoning process in programming tasks tends to be more redundant. DTSR is able to reduce the generation length by over 50% while incurring only minimal performance degradation. Overthinking is particularly pronounced in programming tasks, as they typically involve longer answers, requiring the model to spend more tokens on verification and reflection. In many cases, the model is already capable of generating correct code during the intermediate reasoning steps, repeated verification prolongs this process.

## 4 Analysis and Discussion

### 4.1 Ablation Study

**Performance with different token budget** We expand the maximum generation token budget from

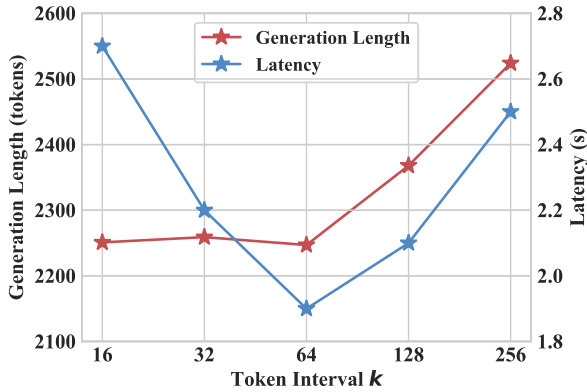


Figure 4: Impact of token interval  $k$ .

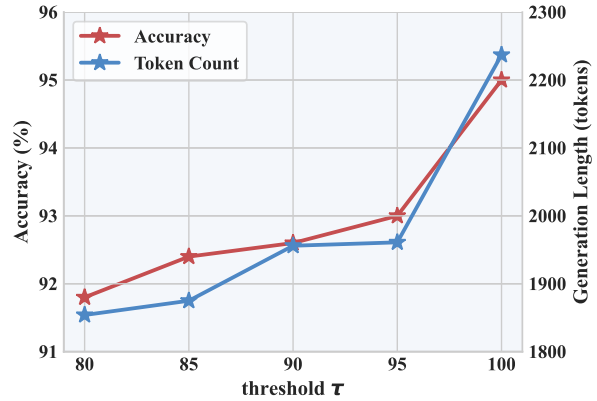


Figure 5: Impact of threshold  $\tau$ .

2k to 16k to compare Vanilla and DTSR in terms of accuracy and generation length. Figure 3 shows that DTSR consistently generates shorter sequences across all tested budgets, with accuracy only showing a slight fluctuation at 16k, while outperforming Vanilla in the other settings. Notably, the difference in sequence length between Vanilla and DTSR grows as the budget increases, suggesting that DTSR achieves greater token savings under larger budgets. In practice, our method can dynamically adjust the token budget to suit diverse application scenarios, thereby improving flexibility and efficiency in real-world deployments.

**Impact of token interval  $k$**  To begin with, varying the token interval  $k$  has almost no effect on accuracy, since the criterion for assessing thought sufficiency remains unchanged. Instead,  $k$  primarily affects two factors: generation length and inference latency. Figure 4 presents the results of DTSR on MATH-500 under different token intervals. When  $k \leq 64$ , the generation length remains nearly constant. However, once  $k > 64$ , the length starts to increase as  $k$  grows. This occurs because overly large token intervals may cause the model to miss the optimal exit points for certain problems, thereby delaying exit. Regarding latency, an overall U-shaped trend can be observed. When  $k \leq 64$ , the reasoning latency increases due to the frequent thought-sufficiency checks required at smaller intervals. In contrast, when  $k > 64$ , although the number of checks is reduced, the model tends to generate longer reasoning traces after missing optimal exit points, which in turn leads to increased latency. Consequently,  $k = 64$  represents a balanced trade-off between generation length and latency, making it a desirable choice in practice.

| Method  | MATH-500    |            | AMC         |            |
|---------|-------------|------------|-------------|------------|
|         | Len         | Latency    | Len         | Latency    |
| Vanilla | 4480        | 3.1        | 6562        | 8.3        |
| DEER    | 2601        | 4.2        | 4514        | 15.6       |
| DTSR    | <b>2247</b> | <b>1.9</b> | <b>3975</b> | <b>6.2</b> |

Table 2: Comparison of Inference Latency (s).

**Impact of threshold  $\tau$**  Figure 5 presents the performance of DTSR on the MATH-500 dataset across different threshold values of  $\tau$ . When the threshold is low, accuracy drops significantly because the model tends to terminate reasoning prematurely, before its thought process is fully developed. This leads to incorrect answers, even though the overall generation length becomes somewhat shorter. These results indicate that the optimal exit point for reasoning is achieved when  $\tau = 100$ , at which the model shows strong confidence in the sufficiency of its reasoning process.

## 4.2 Analysis of Inference Latency

Table 2 reports the end-to-end inference latency of DTSR on the MATH-500 and AMC datasets. The results show that our method not only substantially reduces the generation length, but also achieves a significant reduction in inference latency, yielding improvements of approximately 25%–40%. In contrast, although DEER reduces the final generation length, it unexpectedly increases inference latency. This performance gap can be attributed to two key factors. First, we enable the continuous batching strategy in vLLM, which allows different samples to interleave between the generation and evaluation phases. In contrast, DEER adopts a synchronous scheme that requires all samples to complete generation before a collective evaluation, resulting in

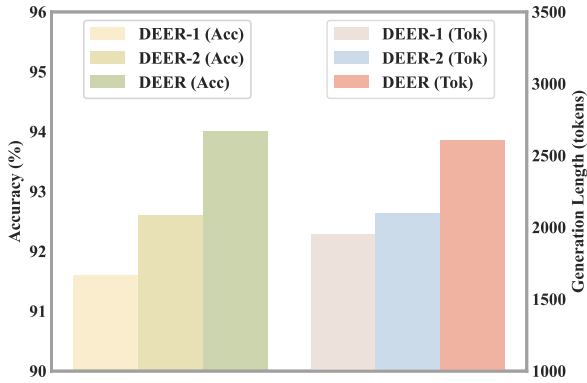


Figure 6: Comparison between DEER and its two variants, DEER-1 and DEER-2.

idle computation and increased latency. Second, we introduce a token interval parameter  $k$ , which substantially reduces redundant checks, thereby further improving overall inference efficiency.

### 4.3 Overconfidence in LRMs

Here, we discuss the issue of overconfidence in LRM. In fact, DEER has observed in experiments that the Qwen-3 series models tend to be overconfident in their confidence predictions. In the original setup, DEER adds an early-exit condition on top of the confidence score derived from intermediate answers: the model is required to generate `</think>` after producing a trial answer. To further investigate this issue, we derive two variants from DEER:

- **DEER-1:** The early-exit decision is based solely on the confidence score of the trial answer.
- **DEER-2:** Whether the model generates `</think>` after producing an intermediate answer is used as the criterion for early exit.

Figure 6 presents the performance of DEER and its two variants, DEER-1 and DEER-2, on MATH-500. DEER-1, which operates without the constraint of generating `</think>`, achieves the lowest accuracy, highlighting the model’s overconfidence in its generated answers. This indicates that relying solely on the confidence score of a trial answer is insufficient for determining when to exit reasoning. In contrast, DEER-2, which uses the simple criterion of whether `</think>` is generated, attains higher accuracy than DEER-1. The generation of `</think>` after producing an answer is a behavior learned during training, serving as a signal of the model’s self-assessment of answer correctness. These results suggest that leveraging the model’s intrinsic self-evaluation is a more reliable approach

| Method                | Accuracy ( $\uparrow$ ) | Length ( $\downarrow$ ) |
|-----------------------|-------------------------|-------------------------|
| DTSR                  | 95.0                    | 2247                    |
| DTSR-1 ( $\tau=100$ ) | 94.2                    | 3254                    |
| DTSR-1 ( $\tau=95$ )  | 93.8                    | 2457                    |

Table 3: Comparison between DTSR and DTSR-1.

for guiding early-exit reasoning than using externally designed criteria.

### 4.4 Self-Evaluation Paradigms

In DTSR, self-evaluation is conducted from a third-person perspective rather than by having the model directly score its own thoughts. The two approaches are illustrated in Figure 9. To compare them, we introduce another variant, DTSR-1, in which an additional prompt is added during the intermediate reasoning process to ask the model to directly evaluate the adequacy of its prior reasoning. The results on MATH-500 are shown in Table 3. It is clear that DTSR-1 performs worse than DTSR in both accuracy and generation length. This is primarily because the model is unable to accurately assess the sufficiency of its reasoning during the process. As the saying goes, “the observer sees clearly what the participant cannot”. Only by evaluating from a third-person perspective and separating reasoning generation from the judgment of its adequacy can the model demonstrate its full reasoning capability.

### 4.5 Case Study

We conduct case studies on three types of problems: mathematical, programming, and open-ended. Figure 10 shows the detailed reasoning process of Qwen3-14B on a math problem. At the beginning, the model generates reasoning normally. When it produces the reflection signal, it starts to evaluate the adequacy of its previous thoughts and gives a score of 75, which is below the early-exit threshold  $\tau$ . As a result, the model returns to the reflection signal position to continue reasoning. In the third round of reasoning, several reflection signals appear, but the model performs the thought sufficiency check only at the last one. This is because the number of generated tokens has not yet reached the token interval  $k$ , and redundant checks are skipped. After several checks, the model finally gives a score of 100 on the fourth attempt, satisfying the early-exit condition. It then appends `</think>` and proceeds to make a conclusion, generating the final answer. Figure 11 shows how

DTSR solves a programming problem, and Figure 12 presents the comparison between Vanilla and DTSR on an open-ended question. Across all cases, DTSR reduces token usage substantially while maintaining performance, improving reasoning efficiency and alleviating excessive thinking.

## 5 Related work

Recently, Large Reasoning Models (LRMs) have demonstrated remarkable performance on complex reasoning tasks by generating long Chains of Thought (CoT) for deep reasoning (Jaech et al., 2024; Guo et al., 2025). However, recent studies (Chen et al., 2025b; Cuadron et al., 2025) have revealed the issue of overthinking, where models continue to reason even after obtaining the correct answer, leading to redundant computational overhead. To address this, prior research has explored a variety of approaches to enable efficient reasoning (Feng et al., 2025), which can be broadly categorized into three directions: (Input) prompt-guided methods, (Model) post-training methods, and (Output) decoding-intervention methods.

**Prompt-guided Methods** Such methods leverage carefully designed prompts to guide the model in performing efficient reasoning. Concise CoT (Renze and Guven, 2024) simply adds the instruction “Be concise” to the prompt, whereas TALE (Han et al., 2025) first estimates an optimal token budget and then incorporates it into the prompt to encourage shorter reasoning chains. Chain-of-Draft (Xu et al., 2025a) constrains the model to preserve only the most essential reasoning drafts at each step, thereby reducing token usage in few-shot settings. Although easy to implement, these approaches (Lee et al., 2025) are limited in compressing long reasoning chains, and therefore better suited for use alongside other methods.

**Post-training Methods** These methods typically employ supervised fine-tuning (SFT) using variable-length CoT data (Yu et al., 2024; Liu et al., 2024; Ma et al., 2025b), or leverage reinforcement learning (RL) with length-based rewards (Team et al., 2025; Arora and Zanette, 2025b; Yeo et al., 2025; Yi et al., 2025). L1 (Aggarwal and Welleck, 2025) and Elastic Reasoning (Xu et al., 2025b) incorporate length constraints into the reward function, training models to flexibly adjust the length of reasoning chains under different token budgets. Recently, numerous studies (Lou et al., 2025; Jiang

et al., 2025b; Fang et al., 2025) have focused on training hybrid reasoning models that adaptively choose between concise responses and long-chain reasoning based on task complexity. While these methods attempt to address the problem at the fundamental level, their high training costs restrict most experiments to smaller models (e.g., 1.5B or 7B), limiting their applicability to larger models.

**Decoding-intervention Methods** Such methods intervene during decoding to adjust generation, guiding models to produce answers earlier (Qiao et al., 2025; Huang et al., 2025) or exit reasoning sooner (Liu and Wang, 2025; Jiang et al., 2025a; Zhang et al., 2025). NoWAIT (Wang et al., 2025) attempts to suppress explicit self-reflection by blocking tokens like Wait and Hmm during reasoning. However, the model can readily bypass this restriction by generating other tokens. Dynasor-CoT (Fu et al., 2025) probes the model’s intermediate answers at fixed token intervals and stops thinking if three consecutive answers are consistent, whereas DEER (Yang et al., 2025b) tracks intermediate answers at reasoning transition points and determines whether to terminate reasoning based on their confidence scores. However, both approaches rely on probing intermediate answers to determine stopping points, which works well only for problems with definite answers, but is less effective for long-form or open-ended outputs. In contrast, our approach effectively overcomes this limitation.

## 6 Conclusion

In this work, we alleviate the inefficiency of LRMs by introducing Dynamic Thought Sufficiency in Reasoning (DTSR), a novel framework that enables models to dynamically determine when their CoT is sufficient to yield a reliable answer. By integrating reflection signal monitoring with thought sufficiency checks, DTSR provides a principled mechanism for mitigating overthinking while maintaining strong reasoning performance. Experiments on Qwen3 models demonstrate that DTSR achieves substantial reductions in reasoning length with only minimal performance degradation, highlighting its effectiveness in improving efficiency. We further discuss overconfidence in LRMs and the self-evaluation paradigm, shedding light on how models effectively assess the sufficiency of their CoT. This exploration offers a fresh perspective and valuable insights for guiding early-exit reasoning.

## 610 **Limitations**

611 While DTSR demonstrates promising results in  
612 improving reasoning efficiency, we identify some  
613 possible limitations as follows.

- 614 • Due to computational resource constraints, we  
615 conduct experiments only on LLMs up to 32B.
- 616 • This work focuses exclusively on textual reason-  
617 ing tasks, including mathematics and code. We  
618 have not yet extended our method to multimodal  
619 reasoning or agent scenarios, which we leave for  
620 future exploration.

## 621 **References**

622 Pranjali Aggarwal and Sean Welleck. 2025. **L1: Con-**  
623 **trolling how long a reasoning model thinks with rein-**  
624 **forcement learning.** *Preprint*, arXiv:2503.04697.

625 Daman Arora and Andrea Zanette. 2025a. **Training**  
626 **language models to reason efficiently.** In *The Thirty-*  
627 *ninth Annual Conference on Neural Information Pro-*  
628 *cessing Systems*.

629 Daman Arora and Andrea Zanette. 2025b. **Train-**  
630 **ing language models to reason efficiently.** *CoRR*,  
631 abs/2502.04463.

632 Paul C. Bogdan, Uzay Macar, Neel Nanda, and Arthur  
633 Conmy. 2025. **Thought anchors: Which llm reason-**  
634 **ing steps matter?** *Preprint*, arXiv:2506.19143.

635 Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng,  
636 Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang  
637 Zhou, Te Gao, and Wanxiang Che. 2025a. **To-**  
638 **wards reasoning era: A survey of long chain-of-**  
639 **thought for reasoning large language models.** *CoRR*,  
640 abs/2503.09567.

641 Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He,  
642 Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi  
643 Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang,  
644 Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025b. **Do**  
645 **NOT think that much for 2+3=? on the overthinking**  
646 **of long reasoning models.** In *Forty-second Interna-*  
647 *tional Conference on Machine Learning*.

648 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian,  
649 Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias  
650 Plappert, Jerry Tworek, Jacob Hilton, Reiichiro  
651 Nakano, Christopher Hesse, and John Schulman.  
652 2021. **Training verifiers to solve math word prob-**  
653 **lems.** *Preprint*, arXiv:2110.14168.

654 Gheorghe Comanici, Eric Bieber, Mike Schaeckermann,  
655 Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Mar-  
656 cel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and  
657 1 others. 2025. **Gemini 2.5: Pushing the frontier with**  
658 **advanced reasoning, multimodality, long context,**  
659 **and next generation agentic capabilities.** *Preprint*,  
660 arXiv:2507.06261.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao  
Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu,  
Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao,  
Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana  
Klimovic, Graham Neubig, and Joseph E. Gonzalez.  
2025. **The danger of overthinking: Examining the**  
**reasoning-action dilemma in agentic tasks.** *Preprint*,  
arXiv:2502.08235.

Mz Dai, Chenxu Yang, and Qingyi Si. 2025. **S-GRPO:**  
**Early exit via reinforcement learning in reasoning**  
**models.** In *The Thirty-ninth Annual Conference on*  
*Neural Information Processing Systems*.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025.  
**Thinkless: Llm learns when to think.** *Preprint*,  
arXiv:2505.13379.

Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao  
Wang. 2025. **Efficient reasoning models: A survey.**  
*Preprint*, arXiv:2504.10903.

John H. Flavell. 1979. **Metacognition and cognitive**  
**monitoring: A new area of cognitive-developmental**  
**inquiry.** *American Psychologist*, 34:906–911.

D. S. Fleur, B. Bredeweg, and W. van den Bos. 2021.  
**Metacognition: Ideas and insights from neuro- and**  
**educational sciences.** *npj Science of Learning*, 6:1–  
11.

Yichao Fu, Junda Chen, Yonghao Zhuang, Zheyu Fu,  
Ion Stoica, and Hao Zhang. 2025. **Reasoning without**  
**self-doubt: More efficient chain-of-thought through**  
**certainty probing.** In *ICLR 2025 Workshop on Founda-*  
*tion Models in the Wild*.

Kanishk Gandhi, Ayush K Chakravarthy, Anikait Singh,  
Nathan Lile, and Noah Goodman. 2025. **Cognitive**  
**behaviors that enable self-improving reasoners, or,**  
**four habits of highly effective STars.** In *Second Con-*  
*ference on Language Modeling*.

D. Guo, D. Yang, H. Zhang, and et al. 2025. **Deepseek-**  
**r1 incentivizes reasoning in llms through reinforce-**  
**ment learning.** *Nature*, 645:633–638.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu  
Zhao, Shiqing Ma, and Zhenyu Chen. 2025. **Token-**  
**budget-aware LLM reasoning.** In *Findings of the As-*  
*sociation for Computational Linguistics: ACL 2025*,  
pages 24842–24855, Vienna, Austria. Association  
for Computational Linguistics.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu,  
Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie  
Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan  
Liu, and Maosong Sun. 2024. **OlympiadBench:**  
**A challenging benchmark for promoting AGI with**  
**olympiad-level bilingual multimodal scientific prob-**  
**lems.** In *Proceedings of the 62nd Annual Meeting of*  
*the Association for Computational Linguistics (Vol-*  
*ume 1: Long Papers)*, pages 3828–3850, Bangkok,  
Thailand. Association for Computational Linguistics.



826 Yuhui Xu, Hanze Dong, Lei Wang, Doyen Sahoo,  
827 Junnan Li, and Caiming Xiong. 2025b. [Scalable](#)  
828 [chain of thoughts via elastic reasoning](#). *Preprint*,  
829 arXiv:2505.05315.

830 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang,  
831 Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao,  
832 Chengen Huang, Chenxu Lv, Chujie Zheng, Day-  
833 iheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao  
834 Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41  
835 others. 2025a. [Qwen3 technical report](#). *Preprint*,  
836 arXiv:2505.09388.

837 Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu,  
838 Chenyu Zhu, Qiaowei Li, Zheng Lin, Li Cao, and  
839 Weiping Wang. 2025b. [Dynamic early exit in reason-](#)  
840 [ing models](#). *Preprint*, arXiv:2504.15895.

841 Edward Yeo, Yuxuan Tong, Xinyao Niu, Graham Neu-  
842 big, and Xiang Yue. 2025. [Demystifying long chain-](#)  
843 [of-thought reasoning in LLMs](#). In *ICLR 2025 Work-*  
844 *shop on Deep Generative Model in Machine Learn-*  
845 *ing: Theory, Principle and Efficacy*.

846 Jingyang Yi, Jiazheng Wang, and Sida Li. 2025. [Short-](#)  
847 [erbetter: Guiding reasoning models to find optimal](#)  
848 [inference length for efficient reasoning](#). *Preprint*,  
849 arXiv:2504.21370.

850 Ping Yu, Jing Xu, Jason E Weston, and Iliia Kulikov.  
851 2024. [Distilling system 2 into system 1](#). In *The*  
852 *First Workshop on System-2 Reasoning at Scale*,  
853 *NeurIPS'24*.

854 Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit  
855 Panda, Jinyang Li, and He He. 2025. [Reasoning](#)  
856 [models know when they're right: Probing hidden](#)  
857 [states for self-verification](#). In *Second Conference on*  
858 *Language Modeling*.

## 859 A Reflection Signal

860 In this paper, we take the following keywords as  
861 signals for LRM reflection:

---

“Wait”, “Alternative”, “Alternatively”  
“But wait”, “But let me”, “But let’s”

---

Table 4: Reflection Signal List

862 **Comparison with NoWAIT** NoWAIT sup-  
863 presses all reflection-related tokens, which can dis-  
864 rupt intrinsic reasoning and reduce accuracy. In  
865 contrast, DTSR preserves normal reasoning and  
866 triggers sufficiency evaluation only upon detecting  
867 reflection signals, achieving significant generation-  
868 length reduction without harming performance.

## 869 B Prompt Template

870 Figure 8 presents the prompt template used for  
871 thought sufficiency check stage. Figure 9 illustrates  
872 the differences between the prompt templates of  
873 DTSR and its variant, DTSR-1.

## 874 C More Experiments

875 **Comparison with Training-based Mehtods** We  
876 further compare DTSR with more baselines.  
877 For training-based methods, we include RL +  
878 Length Penalty (Arora and Zanette, 2025a) and  
879 S-GRPO (Dai et al., 2025). RL + Length Penalty  
880 penalizes overly long correct responses by assign-  
881 ing rewards based on their deviation from the mean  
882 length. S-GRPO uses rule-based outcome rewards  
883 to regulate intermediate reasoning, encouraging  
884 LLMs to generate high-quality thoughts earlier and  
885 to exit once sufficient reasoning is reached, thereby  
886 improving efficiency while maintaining accuracy.  
887 As shown in Table 5, DTSR achieves a generation  
888 length comparable to training-based methods. Al-  
889 though it exhibits slightly lower accuracy due to  
890 the absence of additional training, DTSR remains  
891 competitive while requiring no extra training.

892 **Comparison with Training-free Mehtods** We  
893 also compare DTSR with two recent efficient rea-  
894 soning methods, TALE (Han et al., 2025) and  
895 Chain-of-Draft (CoD) (Xu et al., 2025a). As shown  
896 in Table 6, TALE achieves almost no reduction in  
897 generation length, while CoD substantially reduces  
898 token usage at the cost of a significant accuracy  
899 drop. In contrast, DTSR strikes a better balance  
900 between accuracy and generation length.

901 **Performance under Different Decoding Strate-**  
902 **gies** In the main experiments, we adopt a decod-  
903 ing configuration with temperature set to 0.6 and  
904 top\_p set to 0.95. Table 7 reports the results un-  
905 der greedy decoding. The results show that DTSR  
906 maintains strong performance across different de-  
907 coding strategies, further demonstrating the robust-  
908 ness and applicability of our method.

## 909 D Case Study

910 Figures 10, 11 and 12 present representative exam-  
911 ples of Qwen-14B solving mathematical, program-  
912 ming, and open-ended problems under the DTSR  
913 framework, respectively.

**Question:** Let  $ABC$  be a triangle inscribed in circle  $\omega$ . Let the tangents to  $\omega$  at  $B$  and  $C$  intersect at point  $D$ , and let  $\overline{AD}$  intersect  $\omega$  at  $P$ . If  $AB=5$ ,  $BC=9$ , and  $AC=10$ ,  $AP$  can be written as the form  $\frac{m}{n}$ , where  $m$  and  $n$  are relatively prime integers. Find  $m + n$ .

**Full Thought:**

|  |   |
|--|---|
| <code>&lt;think&gt;</code> Okay, so I need to find the probability that, for three randomly chosen distinct vertices $Q, R, S$ of a regular icosahedron, the distance $d(Q, R)$ is greater than $d(R, S)$ . The answer should be in the form $m/n$ where $m$ and $n$ are coprime, and then find $m + n$ .          | <div style="font-size: 3em;">}</div> <div style="border: 1px solid black; padding: 2px; width: fit-content; margin: 0 auto;">Under Thinking</div> |
| <code>\n&lt;/think&gt;</code> \n\n We are given triangle $ABC$ <code>&lt;omitted&gt;</code> ### Final Answer:\n\n <code>\n\boxed{13}</code> \n\$ <span style="float: right;"><b>✗ False</b></span>   |   |
| <code>\n\n</code> First, I remember that when two tangents are drawn from a point to a circle, they are equal in length. So, $DB = DC$ because $D$ is the intersection of the tangents at $B$ and $C$ . <code>&lt;omitted&gt;</code> Wait, but $A$ is on $\omega$ , so power of $A$ would be zero. Hmm, not sure.  |   |
| <code>\n&lt;/think&gt;</code> \n\n We are given triangle $ABC$ <code>&lt;omitted&gt;</code> ### Final Answer:\n\n <code>\n\boxed{137}</code> \n\$ <span style="float: right;"><b>✗ False</b></span>  |   |
| <code>\n\n</code> Alternatively, since $D$ is the intersection of tangents at $B$ and $C$ <code>&lt;omitted&gt;</code> <code>\n\n</code> Simplify:\n\nFirst, note that $88$ and $22$ have common factor $11$ : $88 = 8 * 11$ , $22 = 2 * 11$ .<br>So $88/22 = 4$ .<br>Therefore, $AP = 4 * (25/13) = 100/13$ .     |   |
| <code>\n&lt;/think&gt;</code> \n\n To solve the problem, we are given <code>&lt;omitted&gt;</code> ### Final Answer:\n\n <code>\n\boxed{197}</code> \n\$ <span style="float: right;"><b>✗ False</b></span>   |   |
| <code>\n\n</code> Wait, this is confusing. Maybe I need to use directed lengths or consider the power of point formula with signs. But if $P$ is beyond $A$ from $D$ , then $DA * DP$ would be $DA * (DA + AP)$ . <code>&lt;omitted&gt;</code> But I need to figure out which is the case.                         | <div style="border: 1px solid black; padding: 2px; width: fit-content; margin: 0 auto;">Optimal Exit Point</div>                                  |
| <code>\n&lt;/think&gt;</code> \n\n To solve the problem, we are given <code>&lt;omitted&gt;</code> ### Final Answer:\n\n <code>\n\boxed{113}</code> \n\$ <span style="float: right;"><b>✓ True</b></span>  |   |
| <code>\n\n</code> Wait, that's nice! So $AP = 100/13$ . Which is already in reduced form since $100$ and $13$ are coprime ( $13$ is prime, doesn't divide $100$ ). Therefore, $m = 100$ , $n = 13$ , $m + n = 113$ .   | <div style="font-size: 3em;">}</div> <div style="border: 1px solid black; padding: 2px; width: fit-content; margin: 0 auto;">Over Thinking</div>  |
| <code>\n&lt;/think&gt;</code> \n\n To solve the problem, we are given <code>&lt;omitted&gt;</code> ### Final Answer:\n\n <code>\n\boxed{113}</code> \n\$ <span style="float: right;"><b>✓ True</b></span>  |   |
| <code>\n\n</code> Wait, but let me check my steps again to make sure.<br>First, we found $t = 88/169$ . Then $AP = t *  \text{direction vector}  = (88/169) * (325/22)$ . Then simplifying:<br>Divide $88$ by $22 = 4$ , and $325$ divided by $169 = 25/13$ . So $4 * (25/13) = 100/13$ . Yes, that seems correct. |   |
| <code>\n&lt;/think&gt;</code> \n\n To solve the problem, we are given <code>&lt;omitted&gt;</code> ### Final Answer:\n\n <code>\n\boxed{113}</code> \n\$ <span style="float: right;"><b>✓ True</b></span>  |   |

Figure 7: An illustrative example showing how we identify the optimal exit point. After each generated sentence, we append `</think>` to force the model to produce an answer. The correctness of the resulting answer is then used to determine whether the current reasoning state is sufficient and corresponds to the optimal exit point.

**Thought Sufficiency Check Prompt Template**

```

<|im_start|>users
Assess the confidence that the provided thought is sufficient to solve the question.
Output only in the format: Confidence: <number>, where <number> is a real value
between 0 and 100. Here, 100 means the thought is fully sufficient to reach the
final answer directly without additional reasoning, and 0 means the thought is
completely insufficient to do so.

### Question
{question}

### Thought
{thought}
<|im_end|>
<|im_start|>assistant
<think>

</think>

Confidence:

```

Figure 8: Thought Sufficiency Check Prompt Template

### DTSR

<|im\_start|>users

Assess the confidence that the provided thought is sufficient to solve the question. Output only in the format: Confidence: <number>, where <number> is a real value between 0 and 100. Here, 100 means the thought is fully sufficient to reach the final answer directly without additional reasoning, and 0 means the thought is completely insufficient to do so.

### Question

{How many positive whole-number divisors does 196 have?}

### Thought

{<think>\nOkay, so I need to figure out how many positive whole-number divisors the number 196 has. Hmm, let me think. I remember that to find the number of divisors of a number, you first have to find its prime factorization. Once you have that, you can use exponents to determine the total number of divisors. Let me try to recall exactly how that works.}

<|im\_end|>

<|im\_start|>assistant

<think>

</think>

Confidence:

### DTSR-1

<|im\_start|>system

Please reason step by step, and put your final answer within `\\boxed{}`.

<|im\_end|>

<|im\_start|>users

How many positive whole-number divisors does 196 have?

<|im\_end|>

<|im\_start|>assistant

<think>\nOkay, so I need to figure out how many positive whole-number divisors the number 196 has. Hmm, let me think. I remember that to find the number of divisors of a number, you first have to find its prime factorization. Once you have that, you can use exponents to determine the total number of divisors. Let me try to recall exactly how that works.

Assess the confidence that the provided thought is sufficient to solve the question. Output only in the format: Confidence: <number>, where <number> is a real value between 0 and 100. Here, 100 means the thought is fully sufficient to reach the final answer directly without additional reasoning, and 0 means the thought is completely insufficient to do so.

Confidence:

Figure 9: Differences between DTSR and its variant DTSR-1.

| Method                     | GSM8K |      | MATH-500 |      | AMC  |      | GPQA |      | Overall |      |
|----------------------------|-------|------|----------|------|------|------|------|------|---------|------|
|                            | Acc↑  | Tok↓ | Acc↑     | Tok↓ | Acc↑ | Tok↓ | Acc↑ | Tok↓ | Acc↑    | Tok↓ |
| <i>Qwen3-14B</i>           |       |      |          |      |      |      |      |      |         |      |
| <i>Vanilla</i>             | 96.2  | 1672 | 95.4     | 4503 | 97.5 | 6562 | 65.7 | 7513 | 88.7    | 5062 |
| <i>RL + Length Penalty</i> | 95.8  | 1090 | 95.8     | 2866 | 97.5 | 5059 | 65.4 | 4949 | 88.6    | 3491 |
| <i>S-GRPO</i>              | 96.3  | 952  | 96.4     | 2652 | 100  | 4538 | 67.6 | 4537 | 90.1    | 3170 |
| <i>DTSR</i>                | 96.2  | 849  | 95.0     | 2247 | 97.5 | 3975 | 66.2 | 5916 | 88.7    | 3246 |

Table 5: Comparison of DTSR with training-based efficient reasoning methods. "Acc" denotes accuracy, and "Tok" denotes token count. ↑ indicates that higher values are better, while ↓ indicates that lower values are better.

| Method           | GSM8K |      | MATH-500 |      | AMC  |      | GPQA |      | Overall     |             |
|------------------|-------|------|----------|------|------|------|------|------|-------------|-------------|
|                  | Acc↑  | Tok↓ | Acc↑     | Tok↓ | Acc↑ | Tok↓ | Acc↑ | Tok↓ | Acc↑        | Tok↓        |
| <i>Qwen3-14B</i> |       |      |          |      |      |      |      |      |             |             |
| <i>Vanilla</i>   | 96.2  | 1672 | 95.4     | 4503 | 97.5 | 6562 | 65.7 | 7513 | 88.7        | 5062        |
| <i>TALE</i>      | 95.7  | 1241 | 94.6     | 4484 | 95.0 | 6623 | 65.6 | 7138 | 87.7        | 4872        |
| <i>CoD</i>       | 85.7  | 648  | 75.2     | 2359 | 72.5 | 4122 | 51.0 | 1177 | 71.1        | 2077        |
| <i>DTSR</i>      | 96.2  | 849  | 95.0     | 2247 | 97.5 | 3975 | 66.2 | 5916 | <b>88.7</b> | <b>3246</b> |

Table 6: Comparison of DTSR with TALE and Chain-of-Draft (CoD). "Acc" denotes accuracy, and "Tok" denotes token count. ↑ indicates that higher values are better, while ↓ indicates that lower values are better.

| Method           | GSM8K |      | MATH-500 |      | AMC  |      | GPQA |      | Olympiad |      | Overall     |             |
|------------------|-------|------|----------|------|------|------|------|------|----------|------|-------------|-------------|
|                  | Acc↑  | Tok↓ | Acc↑     | Tok↓ | Acc↑ | Tok↓ | Acc↑ | Tok↓ | Acc↑     | Tok↓ | Acc↑        | Tok↓        |
| <i>Qwen3-14B</i> |       |      |          |      |      |      |      |      |          |      |             |             |
| <i>Vanilla</i>   | 96.3  | 1676 | 94.8     | 4511 | 97.5 | 6998 | 64.1 | 7382 | 65.6     | 8644 | 83.7        | 5842        |
| <i>DTSR</i>      | 96.1  | 814  | 94.6     | 2296 | 97.5 | 3977 | 65.6 | 6103 | 66.8     | 5967 | <b>84.1</b> | <b>3831</b> |

Table 7: Experimental results under greedy decoding. "Acc" denotes accuracy, and "Tok" denotes token count. ↑ indicates that higher values are better, while ↓ indicates that lower values are better.

**Question**

In a table tennis tournament every participant played every other participant exactly once. Although there were twice as many right-handed players as left-handed players, the number of games won by left-handed players was 40% more than the number of games won by right-handed players. (There were no ties and no ambidextrous players.) What is the total number of games played?

**Response**

<think>\nOkay, so I need to solve this table tennis tournament problem. \n\nFirst, the problem says that every participant played every other participant exactly once. <omitted> \n\nTherefore, total games  $T = G_L + G_R = 1.4G_R + G_R = 2.4G_R$ . Score: 70 ❌

**Alternatively**, since  $T$  is also equal to  $(3L)(3L - 1)/2$ , so maybe I can set up equations here. \n\nBut how do I find  $G_L$  and  $G_R$ ? <omitted> \n\nSo adding them: \n\nConvert all terms to halves: \n\n $(L^2 - L)/2 + (4L^2 - 2L)/2 + (4L^2)/2$  Score: 90 ❌

**Wait**,  $L=3: 3 \bmod 8$ .  $L=8: 0 \bmod 8$ .  $L=11: 3 \bmod 8$  (since  $11-8=3$ ). <omitted> **Wait**, maybe my initial thought is not correct. **But wait**,  $L=3$  and  $L=8$  and  $L=11$ ? \n\n <omitted> So seems that  $L \equiv 0 \bmod 8$  or  $L \equiv 3 \bmod 8$ ? Score: 95 ❌

**Wait** let me check  $L = 3 + 8k$ . Therefore,  $L(L - 3) = (3 + 8k)(8k)$ . <omitted> Let me check  $L = 3 - 8 =$  negative, which is invalid. \n\nSo seems that  $L=3$  is the only possible value. Hence total number of games is 36. Score: 100 ✅

\n</think>\n\nTo determine the total number of games played in the table tennis tournament, we analyze the structure of the competition and the given constraints. \n\n---\n\n <omitted> \n\n---\n\n ✅ **Final Answer** \n\n $\boxed{36}$

*Vanilla's response:* 10596 tokens ➔ *DTSR's response:* 824 tokens

Figure 10: An example illustrating the reasoning process of DTSR.

### Question

You are given a 0-indexed 2D integer array `nums`. Initially, your score is 0. Perform the following operations until the matrix becomes empty:  
From each row in the matrix, select the largest number and remove it. In the case of a tie, it does not matter which number is chosen.  
Identify the highest number amongst all those removed in step 1. Add that number to your score.  
Return the final score.

### Response

**<think>** Okay, let's see. The problem is about processing a 2D matrix in such a way that in each step, we take the largest element from each row, **<omitted>** Then repeat until all rows are empty. Score: 50 ✖

**Wait**, but each row's length might be different? **Wait**, no. **Wait**, the problem says that the matrix is 0-indexed 2D array. But the rows can have different lengths? **<omitted>** But how do we process this? Score: 90 ✖

**Wait**, the process continues until the matrix becomes empty. **<omitted>** Then, for each step, we take the first element of each row, find the max among them, which would take the next element in each row (since the first **Score: 100** moved) ✔

```
\n</think>\n\nclass Solution:\n    def matrixSum(self, nums: List[List[int]]) -> int:\n        current_max = max(current_max, row[col])\n        score += current_max\n        return score
```

Vanilla's response: 10596 tokens



DSTR's response: 824 tokens

Figure 11: An example illustrating the reasoning process of DTSR on a programming problem.

## Case Example: Open Question

### Question:

Suppose gravity on Earth suddenly became half as strong.  
How would human architecture, biology, and transportation adapt over time?

### Vanilla:

<think>...</think>

If Earth's gravity suddenly became half as strong (i.e., gravitational acceleration reduced from 9.8 m/s<sup>2</sup> to 4.9 m/s<sup>2</sup>), the effects on human architecture, biology, and transportation would be profound and multifaceted, with immediate and long-term adaptations across these domains.

#### 1. Architecture

- Reduced Structural Load: Buildings, bridges, and other infrastructure would require significantly less material and reinforcement. For example, skyscrapers could be taller or constructed with lighter materials (e.g., advanced composites instead of steel) due to decreased compressive forces.
- Design Adjustments: Foundations might be smaller or less deep, and load-bearing walls could be thinner. However, engineers would need to account for new challenges, such as wind forces (which are less mitigated by gravity) and seismic activity (potentially altered by changes in Earth's mass or tectonic dynamics).
- Fluid Dynamics: Plumbing systems might require adjustments to handle reduced pressure in water distribution, and drainage systems could become less efficient due to slower fluid flow under lower gravity...

#### 2. Human Biology

- Reduced Physical Strain: Humans would feel lighter, making movement easier and reducing joint stress. People might jump higher, run faster, and experience less fatigue in daily activities.
- Musculoskeletal Changes: Over time, muscle atrophy and bone density loss could occur due to reduced mechanical loading. This would mirror the effects of prolonged microgravity on astronauts, potentially leading to osteoporosis and weakened ligaments.
- Cardiovascular Adaptation: The heart would pump less forcefully, and blood pressure might decrease. Circulatory systems might adjust to reduced gravitational pull, potentially altering the distribution of fluids in the body...

#### 3. Transportation

- Easier Space Travel: Launching spacecraft would require less fuel to escape Earth's gravity, making interplanetary missions more feasible and cost-effective.
- Vehicle Design: Cars and other vehicles might become lighter and more agile, with reduced traction requirements...

### DTSR:

<think>...</think>

If Earth's gravity suddenly became half as strong, the following adaptations would occur in architecture, biology, and transportation over time:

#### 1. Architecture

- Structural Design: Buildings would require less material and reinforcement due to reduced gravitational load. Foundations could be shallower, and skyscrapers might reach greater heights since the structural stress is halved.
- Material Use: Lighter materials (e.g., composites, expanded metals) might replace heavier ones like concrete and steel. Open spaces, larger windows, and more cantilevers could become common.
- Wind Considerations: While gravity is lower, wind forces on tall structures might remain a limiting factor, potentially leading to more aerodynamic designs.

#### 2. Biology

- Musculoskeletal System: Humans would likely experience reduced bone density and muscle mass due to less gravitational stress. Over generations, evolutionary adaptations might include taller, more slender bodies with less robust bones and muscles.
- Cardiovascular System: The heart might become smaller, as it would not need to work as hard to pump blood against gravity. However, if atmospheric pressure decreases (due to lower gravity), respiratory adaptations (e.g., larger lungs, higher hemoglobin levels) could emerge to compensate for lower oxygen availability.
- Fluid Distribution: Fluids (e.g., blood, interstitial fluid) would pool more in the upper body, leading to facial puffiness and potential changes in circulatory dynamics.

#### 3. Transportation

- Ground Vehicles: Vehicles would weigh less, improving fuel efficiency and acceleration. Braking systems might need redesigning due to reduced friction (normal force is halved). Heavier cargo could be transported with the same infrastructure.
- Aircraft: If atmospheric pressure remains constant, aircraft would require less lift, enabling smaller, more efficient designs. However, if the atmosphere thins (due to lower gravity), air density would decrease, complicating lift generation. This might necessitate larger wings or more powerful engines.
- Space Travel: Launching to space would require less energy, making interplanetary travel more feasible..

Figure 12: Comparison of Vanilla's and DTSR's answer on an open question.