DTS: Enhancing Large Reasoning Models via Decoding Tree Sketching

Zicheng Xu*
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
zxu161@jh.edu

Guanchu Wang*†
Department of Computer Science
University of North Carolina at Charlotte
Charlotte, NC 28223
gwang16@charlotte.edu

Yu-Neng Chuang
Department of Computer Science
Rice University
Houston, TX 77005
yc146@rice.edu

Guangyao Zheng
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
gzheng7@jhu.edu

Alexander S. Szalay
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
szalay@jhu.edu

Zirui Liu
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455
zrliu@umn.edu

Vladimir Braverman†
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
vova@cs.jhu.edu

Abstract

Large Reasoning Models (LRMs) demonstrate strong performance on complex reasoning tasks, yet they often suffer from **overthinking**—producing excessively long chain-of-thought (CoT) traces that increase inference cost and may degrade accuracy. Our analysis reveals a clear anti-correlation between reasoning length and accuracy, where across multiple stochastic decodes, the short reasoning paths consistently achieve the highest correctness, while longer ones accumulate errors and repetitions. These short optimal reasoning paths can be found ideally through full enumeration of the reasoning space. However, the tree-structured reasoning space grows exponentially with sequence length, rendering exhaustive exploration infeasible. To address this, we propose `DTS`, a model-agnostic decoding framework that sketches the reasoning space by selectively branching at high-entropy tokens and applies early stopping to select the shortest completed reasoning path. This approach approximates the optimal solution that enhances both efficiency and accuracy, without requiring additional training or supervision. Experiments on AIME2024 and AIME2025 datasets with DeepSeek-R1-Distill-Qwen-7B and 1.5B show that `DTS` improves accuracy by up to 8%, reduces average reasoning length by 23%, and decreases repetition frequency by 12%, demonstrating `DTS`'s ability for scalable and efficient LRM reasoning.

*Equal Contribution
†Correspondence to Vladimir Braverman and Guanchu Wang.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Efficient Reasoning.

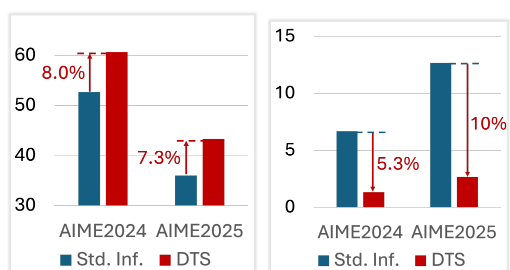
1 Introduction

Large Language Models (LLMs)[27, 2, 4] have demonstrated impressive reasoning capabilities across domains such as mathematics, programming, and scientific problem-solving [23, 28, 10]. Recent progress in reasoning-focused LLMs, often referred to as Large Reasoning Models (LRMs) such as Deepseek R1[7] and OpenAI o1[21], has been driven by supervised fine-tuning (SFT)[22] and reinforcement learning (RL)[3], which encourage step-by-step, CoT reasoning[29, 6]. While such structured reasoning often improves performance on challenging tasks[13, 37], it introduces a major drawback: reasoning models frequently produce overly long responses that ultimately lead to an incorrect response, a phenomenon recent studies refer to as **overthinking** [25, 5].

The overthinking problem leads to two central challenges. First, excessive reasoning chains substantially increase inference latency, limiting the practicality of LRMs in real-world applications[16]. Second, over-reasoning often reduces accuracy rather than improving it, as longer chains tend to accumulate errors or diverge into irrelevant details [24, 33, 17]. To mitigate these issues, recent works on adaptive thinking and CoT pruning, such as AdaptThink[36], AutoL2S[16], and O1-Pruner[17], have explored dynamically adjusting the depth and length of reasoning traces through fine-tuning on long and short CoT data. Others, including Chain of Draft[31], ThinkPrune[9], and AdaCoT[15], attempt to balance reasoning efficiency with accuracy, showing that carefully pruning or adaptively triggering CoT can sometimes preserve or even improve performance. However, these approaches are not robust: aggressive pruning or omitting critical steps often sacrifices accuracy[11], highlighting the difficulty of reliably balancing efficiency with correctness.

Nevertheless, optimizing LRMs for efficient and reliable reasoning remains an open challenge. Most existing approaches reduce overthinking through additional training, such as SFT or RL, which requires resources and labeled data, limiting scalability and accessibility in practice. Moreover, such additional training is not strictly necessary, since base LRMs inherently generate diverse reasoning trajectories during inference. Our analysis in Section 2.2 shows a strong **anti-correlation** between response length and performance, indicating that shorter reasoning paths already embedded within the model tend to yield more accurate solutions. These reasoning trajectories can be naturally organized into a tree-structured reasoning space, where each node represents a generated token and each path corresponds to a complete chain of thought. Conceptually, an oracle capable of exhaustively enumerating and evaluating all paths within this tree would identify the shortest high-performing trajectory, achieving the optimal balance between accuracy and efficiency. However, the exponential growth of this tree-structured reasoning space produces an infinite search space, making exhaustive search computationally infeasible. To address this challenge, we propose to **sketch** the reasoning space into a compact backbone that captures essential branches and approximates the shortest high-performing reasoning path.

We propose `DTS` (Decoding Tree Sketching), a model-agnostic decoding framework that mitigates overthinking by constructing a dynamic reasoning tree at inference time. Specifically, `DTS` leverages next-token entropy to locate critical divergence points, guiding the selective expansion of branches that form the backbone of the reasoning tree. All branches are decoded in parallel through auto-regressive generation, and `DTS` adopts an early-stopping strategy that returns the first completed branch, which corresponds to the shortest reasoning path and serves as an approximation of the shortest optimal trajectory. This design directly aligns with our empirical observation that shorter reasoning paths tend to achieve higher accuracy, enabling `DTS` to balance efficiency and correctness through concise reasoning. Together, these mechanisms allow `DTS` to efficiently traverse the reasoning space, yielding an approximate solution comparable to that obtained through exhaustive enumeration. We evaluate `DTS` on two LRMs across two reasoning benchmarks. As demonstrated in Figure 1, `DTS`



(a) Accuracy ↑ (b) Repetition rate ↓

Figure 1: `DTS` effectively improves accuracy by 8% and 7.3%, and reduce repetition rate by 5.3% and 10% on the AIME2024 and AIME2025, respectively.

attains accuracy improvements of up to 8% and reduces repetition frequency up to 10%. Our contributions are summarized as follows:

- **Training-free and model-agnostic design:** DTS requires no RL or SFT, operates entirely at decoding time, and serves as a plug-and-play module.
- **Improving reasoning process:** DTS naturally reduces overthinking and limits compounding errors by choosing the most information-dense and concise reasoning path.
- **Efficiency:** DTS leverages GPU parallelism to efficiently expand the decoding tree, enabling fast and scalable optimization of reasoning paths.
- **Evaluation:** We validate DTS on two LRMs and two reasoning benchmarks, showing consistent gains in both accuracy and efficiency.

2 Preliminary

2.1 Notations

We consider Large Reasoning Models (LRM) f in this work. For each input prompt x, the reasoning sequence is progressively generated by $\xi_{t+1} = \xi_t \oplus v_t$, and $v_t \sim f(x_t, \xi_t)$ for $t = 1, 2, \cdots$, where $\xi_t \oplus v_j$ denotes to append token v_j to sequence ξ, and the initial output sequence is defined as $\xi_0 = \varnothing$. The generation process is also called auto-regressive generation.

2.2 Analysis of Overthinking

To approximate the aforementioned oracle that evaluates all trajectories in the reasoning space, we run 100 stochastic decodes for each AIME24 problem using DeepSeek-R1-Distill-Qwen-7B [7]. We evaluate the reasoning space in terms of response length for efficiency and accuracy for performance.

Strategy	Acc (%)	Len
Shortest	76.67	4469
Longest	10.00	14919
Overall Mean	51.03	10265

Table 1: Accuracy and average response length per question on AIME24 under different response-length selection strategies.

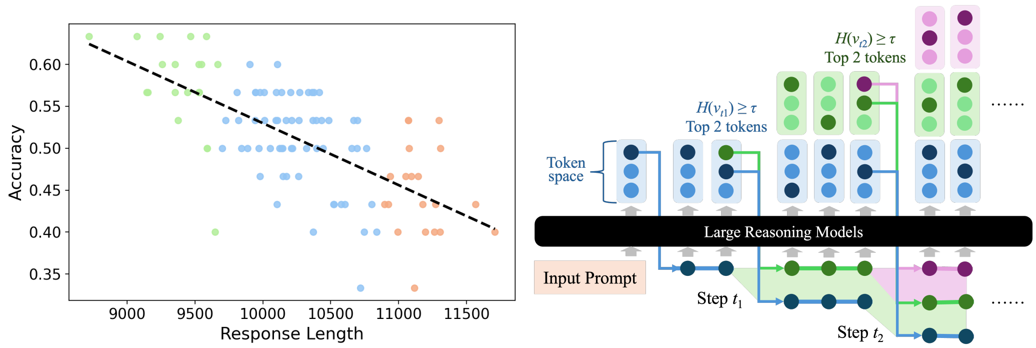
Overthinking vs. Optimal Table 1 shows the performance of the LRM by selecting, for each question, the shortest and the longest response among the 100 samples. Choosing the shortest attains 76.67% accuracy, substantially higher than 10.00% for the longest, and also surpasses the overall mean of 51.03%. This indicates that the reasoning space contains solutions that achieve a better balance between performance and efficiency, and that simple enumeration over multiple trajectories can expose such solutions. Furthermore, it illustrates that reasoning verbosity can degrade quality.

Anti-correlation between Accuracy and Length To further investigate the relationship of LRM's performance in terms of response length, we give Figure 2 (a), where each point represents a single inference run. The green left cluster marks an empirically favorable regime that balances performance and efficiency, whereas the red right cluster corresponds to the overthinking regime. There is a clear anti-correlation with accuracy decreases as response length increases, as shown in Figure 2 (a), indicating that longer reasoning chains are less reliable. These empirical results motivate our DTS objective to improve LRM's accuracy by reducing the reasoning length.

2.3 Chasing Shortest Reasoning Path by Decoding Tree

We follow the observed "Anti-correlation" between the accuracy and reasoning length to optimize the reasoning process. To represent the reasoning space, all possible reasoning paths of an LRM can be naturally represented as a tree structure, where each node corresponds to a possible token in the generated sequence. Starting from the first token, every step in the reasoning process branches into $|\mathbb{T}|$ possible continuations, where $\mathbb{T}$ denotes the token space. In the following steps, the second token expands into $|\mathbb{T}|^2$ possible paths, the third token into $|\mathbb{T}|^3$, and so forth, leading to an exponentially growing tree space $|\mathbb{T}| + |\mathbb{T}|^2 + |\mathbb{T}|^3 + \cdots$.

According to the anti-correlation phenomenon, shorter reasoning paths generally achieve higher accuracy, suggesting that the **optimal solution lies in identifying the shortest path from the root**



(a) (b)

Figure 2: (a) Accuracy vs. response length with a linear regression fit. Each dot represents a single inference run. (b) Generation of the decoding tree by DTS. DTS expands new branches whenever the next-token entropy satisfies $H(v) \geq \tau$.

token to a leaf node. However, the exponential explosion of the reasoning tree produces an effectively infinite search space, making it computationally infeasible to exhaustively traverse every path for finding the globally optimal reasoning path. To this end, our DTS can effectively prune the search space by sketching the growing tree space during decoding, and approximates the optimal solution by identifying the shortest reasoning paths.

3 Decoding Tree Sketching (DTS)

We introduce DTS to sketch the decoding tree to efficiently achieve the shortest reasoning path. The overall framework of DTS is illustrated in Figure 2 (b). Instead of generating new branches at each step, DTS selectively expands branches only at tokens whose next-token distribution exhibits high entropy. Formally, tokens such as v_{t_1} and v_{t_2} in Figure 2 (b) are identified as branch points, while low-entropy tokens continue existing sequences without expansion. Since the variance of the generated output is predominantly determined by high-uncertainty tokens, this selective branching strategy allows DTS to capture the essential backbones of the decoding tree.

3.1 Decoding Tree Generation

The decoding tree in DTS is constructed by three key components: new branch generation, branch-wise auto-regressive generation, and early stopping criteria. These components together define how the tree grows, balances computational efficiency, and determines when to terminate generation. We describe each component in detail below.

New Branch Generation. Unlike standard auto-regressive decoding, which generates a single token at each step, DTS introduces a branch function $F(\cdot)$ to adaptively decide whether to expand multiple branches or continue with a single token. Specifically, when the entropy of the next-token distribution exceeds a threshold τ, $F(\cdot)$ selects the top-K tokens with the highest probabilities to spawn new branches. Otherwise, it samples a single token according to the distribution $P(v)$. This adaptive generation process is illustrated in Figure 2 (b). Formally, given an input prompt x and an intermediate reasoning sequence ξ, the branch function is defined as

$$F(x, \xi) = \begin{cases} \{v_1, \cdots, v_K \mid p_{v_1}, \cdots, p_{v_K} \geq \tilde{p}_K\} & \text{if } H(v) \geq \tau \\ \{v_1\}, v_1 \sim P(v) & \text{if } H(v) < \tau, \end{cases} \tag{1}$$

where $P(v) = f(x, \xi)$ denotes the next-token distribution determined by the LRM; $H(v) = -\sum_{p_v \in P(v)} p_v \log p_v$ estimates the entropy; and $\tilde{p}_K$ is the K-th largest probability in $P(v)$. This mechanism allows DTS to branch out when the prediction is uncertain (high entropy $H(v) \geq \tau$) while conserving space when the prediction is confident (low entropy $H(v) < \tau$). The threshold τ controls the tradeoff between computational breadth and efficiency. In the extreme case $\tau \to +\infty$, DTS reduces to standard auto-regressive decoding with a single token generated at each step.

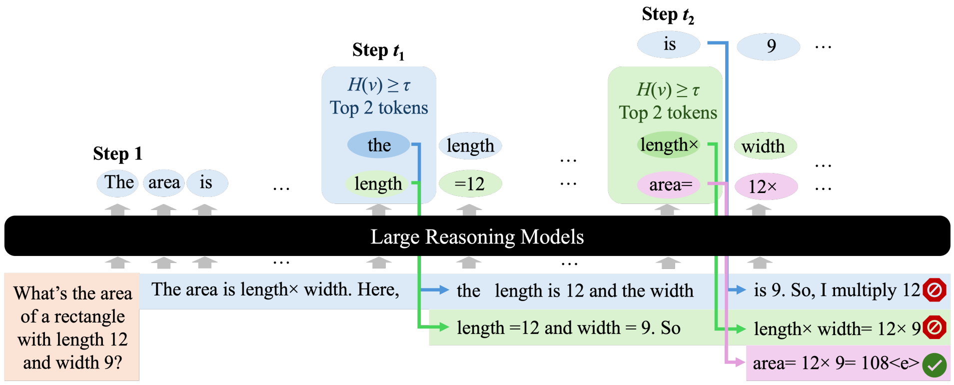


Figure 3: An example of DTS decoding process, given the input prompt 'What's the area of a rectangle with length 12 and width 9?'. DTS generates new branches at steps t_1 and t_2, and stops as soon as any branch terminates with an ending token. The final output is 'The area is length× width. Here, length =12 and width = 9. So area= 12× 9= 108'.

Branch-wise Auto-regressive Generation. DTS performs auto-regressive generation across all branches in parallel, as illustrated in Figure 2 (b). At each time step t, DTS maintains a set of reasoning sequences $\mathcal{T}_t = \{\xi_1, \xi_2, \dots\}$, initialized with $\mathcal{T}_0 = \varnothing$. For every sequence $\xi_i \in \mathcal{T}_t$, the model generates the next tokens based on the branch function $F(x, \xi_i)$ from Equation (1), and appends them to form new sequences $\{\xi_i \oplus v_j \mid v_j \in F(x, \xi_i)\}$. Consequently, the reasoning set is updated as

$$\mathcal{T}_{t+1} = \{\xi \oplus v_j \mid v_j \in F(x, \xi), \xi \in \mathcal{T}_t\}. \tag{2}$$

This process iterates for $t = 0, 1, 2, \dots$, progressively expanding all branches following Equation (2).

Early Stop. DTS terminates the tree growing is motivated by our preliminary results in Figure 2 (a), where short reasoning process consistently outperforms long reasoning process. Following this intuition, DTS selects the shortest sequences as the final reasoning and answer to maximize the performance. Equivalently, DTS stops as soon as any candidate sequence terminates with an ending token, and that completed sequence is returned as the final reasoning and answer. Formally, let `<e>` denotes the ending token[3]. At each time step t, DTS determines an early stop by $\bigvee_{\xi \in \mathcal{T}_t} \mathbb{1}[\texttt{<e>} \in \xi]$, where $\mathbb{1}[\texttt{<e>} \in \xi]$ returns 1 if $\texttt{<e>} \in \xi$ and 0 otherwise. The operator $\bigvee_{\xi \in \mathcal{T}_t}$ aggregates these indicators using logical OR over all reasoning paths $\xi \in \mathcal{T}_t$.

An Example. An illustrative example of DTS is shown in Figure 3. Given the input prompt: 'What's the area of a rectangle with length 12 and width 9?' From step 1 to $t_1 - 1$, the next-token entropy satisfies $H(v) < \tau$, thus DTS samples a single token per step to produce the prefix 'The area is length× width.' (blue). At step t_1, after feeding the token 'Here,' into the model, the next-token entropy $H(v) \geq \tau$; DTS therefore generates new branches (green) by selecting the top two tokens 'the' and 'length', where each branch starts with these two tokens. From step $t_1 + 1$ to $t_2 - 1$, DTS proceeds each branch (blue and branch) with single-token sampling due to $H(v) < \tau$. At step t_2, after feeding the token 'So', the condition $H(v) \geq \tau$ holds again, and DTS expands new branches by selecting the top two tokens 'length×' and 'area=', yielding three branches in total (blue, green purple). After step t_2, decoding continues with single-token sampling along all active branches and stops as soon as any branch emits the end token `<e>` (purple branch). The final output is 'The area is length× width. Here, length =12 and width = 9. So area= 12× 9= 108'.

3.2 The Algorithm of DTS

DTS is described in Algorithm 1. The algorithm begins by initializing the reasoning set with $\varnothing$ (line 1). During the decoding process, DTS follows Equation (1) to expands new branches (line 3); and then follows Equation (2) to update the reasoning set (line 4). The decoding process stops as soon as any reasoning path terminates with the ending token (line 6), where that ended sequence is returned as

[3] depends on the tokenizer

the final reasoning and answer (line 7). Overall, DTS follows a breadth-first Search strategy over the sketched decoding tree, ensuring the shortest reasoning path is identified. All reasoning paths are expanded in parallel by leveraging GPU parallelism, which ensures both the efficiency and scalability of DTS.

4 Experiments

In this section, we conduct experiments to evaluate the performance of DTS framework, aiming to answer the following research questions: **RQ1**: How does DTS perform on LRM reasoning tasks in terms of accuracy and efficiency? **RQ2**: Can DTS mitigate the repetition problem during reasoning generation? **RQ3**: What does the reasoning process produced by DTS look like?

Algorithm 1 Decoding Tree Sketching (DTS)

Input: LRM f, input prompt x.
Output: Optimal reasoning process ξ^*.

1: $\mathcal{T}_0 = \varnothing$
2: **for** $t = 1, 2, \cdots$ **do**
3:     Generate new branches by Eq (1)
4:     Collect sequence candidates $\mathcal{T}_t$ by Eq (2)
5:     **if** $\bigvee_{\xi \in \mathcal{T}_t} \mathbb{1}[\texttt{<e>} \in \xi]$ **then**
6:         **return** $\xi^* = \arg_{\xi \in \mathcal{T}_t} \max \mathbb{1}[\texttt{<e>} \in \xi]$
7:     **end if**
8: **end for**

4.1 Experimental Setup

We specify the models, datasets, evaluation metrics, and implementation details below.

Models. We evaluate DTS using two popular LRMs: Deepseek-R1-Distill-Qwen-7B and Deepseek-R1-Distill-Qwen-1.5B [7], loading their pre-trained weights from Huggingface [30].

Datasets. The evaluation of DTS is based on the AIME24 [19] and AIME25 [18] datasets. Each dataset consists of 30 challenging mathematical problems from the American Invitational Mathematics Examination (AIME), a well-established benchmark for testing the advanced problem-solving and reasoning abilities of LRMs. Following prior work, we use the official problem statements without modification and evaluate model outputs against the unique integer solutions provided by the exam.

Metrics. We control stochasticity by fixing the random seed $s \in \{0, 1, 2, 3, 4\}$ and report mean accuracy over these five runs. In addition to accuracy, we measure efficiency by reporting the average response length generated per problem.

Implementation Details. For all experiments, the maximum generation length is set to each model's maximum token capacity. We use a decoding temperature of 0.6 on AIME24 and 0.5 on AIME25 for both standard inference and DTS. For DTS, we set $K = 3$, $\tau = 2.5$. All models and datasets are accessed through the HuggingFace Transformers [30] and Datasets [14] library.

4.2 Accuracy and Efficiency Improvement (RQ1)

We provide Table 2 to summarize the accuracy and efficiency comparison between our proposed DTS and standard inference across AIME2024 and AIME2025. For both DeepSeek-R1-Distill-Qwen-7B and 1.5B models, DTS consistently improves accuracy while substantially reducing response length. Specifically, on the 7B model, DTS achieves an average accuracy gain of +7.66% and a 22.96% reduction in response length compared to standard inference. Similarly, on the 1.5B model, DTS yields a +4.00% increase in accuracy and a 23.72% reduction in length. These results demonstrate that DTS effectively mitigates overthinking and generates more concise and accurate responses, balancing performance and efficiency without any training involved.

4.3 Reduction of Repetitive Reasoning (RQ2)

A critical challenge in LRMs' reasoning efficiency is the endless repetition problem [34], a special case of overthinking where the model falls into a reasoning loop and continuously generates repeating phrases or tokens without reaching a conclusion. Such repetition prevents the model from completing reasoning traces within the maximum token limit. The problem is particularly severe, as it increases

	AIME2024		AIME2025		Average	
	Acc (%)	Len	Acc (%)	Len	Acc (%)	Len
DeepSeek-R1-Distill-Qwen-7B						
Standard Inference	52.67	13902	36.00	15053	44.34	14478
DTS	**60.67** (+8.00%)	**9865** (-29.03%)	**43.33** (+7.33%)	**12440** (-17.35%)	**52.00** (+7.66%)	**11153** (-22.96%)
DeepSeek-R1-Distill-Qwen-1.5B						
Standard Inference	26.67	16596	24.67	17809	25.67	17203
DTS	**32.67** (+6.00%)	**12462** (-24.91%)	**26.67** (+2.00%)	**13762** (-22.72%)	**29.67** (+4.00%)	**13112** (-23.72%)

Table 2: Average Accuracy (Acc) and response length (Len) on AIME2024 and AIME2025 for DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Qwen-1.5B models. Relative changes of DTS compared to standard inference are shown in parentheses.

both inference time and memory usage while providing no additional reasoning benefit. DTS provides a solution to this failure mode intrinsically by sketching the reasoning space and favoring the shortest completed trajectory. Paths that begin to repeat are overridden by concise completions or pruned through the process, preventing the decoder from drifting into repeated segments.

Table 3 reports the rate of cases where endless repetition occurred under standard inference and our proposed DTS method. Across both AIME2024 and AIME2025 benchmarks, and for both 7B and 1.5B model scales, DTS consistently reduces the occurrence of repetition. On AIME2024, the repetition frequency drops from 6.7% to 1.3% for the 7B model and from 15.3% to 4.7% for the 1.5B model. On AIME2025, the reduction is from 12.7% to 2.7% for the 7B model and from 26.7% to 6.0% for the 1.5B model. These results confirm that DTS substantially mitigates endless repetition, leading to more stable and efficient reasoning trajectories.

	AIME2024		AIME2025	
	7B	1.5B	7B	1.5B
Std. Inf.	6.7%	15.3%	12.7%	26.7%
DTS	**1.3%**	**4.7%**	**2.7%**	**6.0%**

Table 3: Rate (%) of endless repetition cases $\downarrow$ under standard inference and DTS on AIME2024 and AIME2025 for DeepSeek-R1-Distill-Qwen-7B and 1.5B models.

4.4 Case Study on AIME2024 (RQ3)

To further illustrate how DTS improves reasoning quality and efficiency, we present a case study in Figure 4 from the AIME2024 dataset using the DeepSeek-R1-Distill-Qwen-7B model.

Under standard inference, the model exhibits overthinking. It generates excessively long reasoning chains by repeatedly attempting similar calculations and fails to converge to a final solution after reaching the maximum token limit of the model. This is an example of repetitive reasoning discussed in Section 4.3, where the repetition is highlighted in red. In contrast, DTS quickly sketches toward a correct and short reasoning trajectory, arriving at the correct answer using only 3,195 tokens, demonstrating both improved efficiency and correctness.

5 Related Work

Large Reasoning Models Recent frontier large reasoning models, such as OpenAI o1 [21], DeepSeek-R1 [7], and QwQ [32], are specifically optimized to simulate human-like multi-step thinking, producing step-by-step chains of thought that enable them to tackle challenging domains such as mathematical proofs and algorithmic reasoning. Their reasoning competence is typically shaped through large-scale RL with verified reward or by fine-tuning on curated datasets containing explicit reasoning traces.

Question: Alice chooses a set A of positive integers. Then Bob lists all finite nonempty sets B of positive integers with the property that the maximum element of B belongs to A. Bob's list has 2024 sets. Find the sum of the elements of A.
Correct Answer: 55

DTS Response: Alright, so I have this problem here where Alice chooses a set \(A \) of positive integers…So, mathematically, \(\sum_{a \in A} 2^{a-1} = 2024 \)…Dividing 2024 by 2: 2024 /2 = 1012, remainder 0…1 /2 = 0, remainder 1. So writing the remainders in reverse: 11111101000…So, 2024 = 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^3. So, for the exponents 10,9,8,7,6,5,3, that would mean \(a = 11,10,9,8,7,6,4 \)…Thus, the sum of the elements of \(A \) is \(\boxed{55}\). **[3195 Tokens]**
Answer: **55**

Standard Inference Response: Alright, so I have this problem here where Alice chooses a set A of positive integers, and then Bob lists all finite nonempty sets B of positive integers such that the maximum element of B is in A… So, we can try to find such k. So, let's compute 2024 +k and count the number of ones in binary. Let's start with small k. k=1: 2025. 2025 in binary: 11111101001, which has 8 ones. Not equal to k=1. k=2:2026. 2026 in binary:11111101010, which has 8 ones. Not equal to k=2… k=399:2423. 2423 in binary:100100001011, which has 4 ones. k=400:2424. 2424 in binary:100100011000, which has 4 ones… **[32678 Tokens]**
Answer: No conclusion reached due to overthinking

Figure 4: A case study illustrating how DTS produces a concise and correct solution, while standard inference overthinks and enters an endless repetition loop, failing to reach a conclusion after consuming the maximum 32,678 tokens.

Efficient Reasoning While LRM enhances performance through reasoning, it also leads to excessive computation and latency with overthinking [25, 5]. To address this, recent research on efficient reasoning focuses on generating concise yet effective reasoning trajectories without compromising solution quality. Existing approaches can be broadly categorized into two directions. First, RL methods integrate length-aware objectives into reward functions, encouraging models to favor shorter, correct solutions while penalizing unnecessarily verbose reasoning. For example, O1-Pruner [17] introduces a length-harmonizing reward that compares the predicted CoT length to a reference while enforcing accuracy constraints. Kimi K1.5 injects a direct length penalty into its policy optimization to control reasoning depth [26]. L1 conditions training on "Think for N tokens" and penalizes deviation from the target length under group relative policy optimization (GRPO) [1]. These RL methods enable models to internalize efficiency during training and maintain strong reasoning performance. Second, SFT approaches construct variable-length CoT datasets containing both long and short reasoning paths. For instance, C3oT [12] compresses long chains of thought into concise yet faithful traces using a stronger teacher, and the compressed outputs are collected as a short-CoT corpus for supervised fine-tuning. AutoL2S [16] pairs long and short form CoT reasoning paths with <EASY> tokens as an SFT dataset, allowing models to switch to short form reasoning for easier questions. By fine-tuning on such curated data, models learn to produce compact reasoning traces that preserve essential logical steps while avoiding redundancy. However, both of these directions require fine-tuning, limiting accessibility with labeled datasets and training resources. While trainless methods do exist, such as Chain of Draft [31] and CCoT [20] prompting, they do not achieve stable performance gains while maintaining low response length, motivating a reliable training-free efficient reasoning method.

Tree-based Decoding Scheme Tree-based decoding expands inference beyond a single left-to-right trajectory by explicitly exploring a search tree over intermediate states. This decoding scheme has been critical to natural language processing research (NLP), exemplified by the beam search algorithm. Recent studies use tree structures to guide LLM inference paths and enhance LLM's performance. For instance, Tree-of-Thoughts [35] organizes candidate reasoning steps into a tree and conducts lookahead and backtracking with breadth and depth first search to select promising branches. Reasoning via Planning [8] frames reasoning as planning and couples an LLM "world model" with Monte Carlo Tree Search (MCTS) to select high-reward paths. Language Agent Tree Search (LATS) unifies reasoning, tool use, and acting through MCTS. Collectively, these approaches trade added test-time compute for higher reliability and controllability by searching over structured reasoning spaces rather than committing to a single chain.

6 Conclusion

In this work, we introduced DTS, a training-free and model-agnostic decoding framework that enhances both reasoning performance and efficiency for LRMs. By selectively branching at high-entropy tokens and employing early stopping based on the shortest completed path, DTS effectively sketches the reasoning space to approximate an oracle search without exhaustive enumeration. Our empirical results on two reasoning benchmarks, AIME2024 and AIME2025, demonstrate that DTS consistently outperforms standard inference, improving accuracy by up to 8% while reducing reasoning length by more than 20%. Furthermore, DTS significantly mitigates the problem of endless repetition, leading to more concise and reliable reasoning trajectories.

Acknowledgments

This research was supported by NSF CNS 2528780. We would like to thank the Center for Language and Speech Processing at Johns Hopkins University for providing hardware and technical assistance for experiments. The views and conclusions in this paper are those of the authors and do not represent the views of any funding or supporting agencies.

References

[1] Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

[2] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[3] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[4] Chia-Yuan Chang, Zhimeng Jiang, Vineeth Rakesh, Menghai Pan, Chin-Chia Michael Yeh, Guanchu Wang, Mingzhi Hu, Zhichao Xu, Yan Zheng, Mahashweta Das, et al. Main-rag: Multi-agent filtering retrieval-augmented generation. *arXiv preprint arXiv:2501.00332*, 2024.

[5] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

[6] Yu-Neng Chuang, Leisheng Yu, Guanchu Wang, Lizhe Zhang, Zirui Liu, Xuanting Cai, Yang Sui, Vladimir Braverman, and Xia Hu. Confident or seek stronger: Exploring uncertainty-based on-device llm routing from benchmarking to generalization. *arXiv preprint arXiv:2502.04428*, 2025.

[7] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[8] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

[9] Yifan Hou, Zhen Wang, Mengdi Li, and Jianfeng Gao. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*, 2025.

[10] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*, 2024.

[11] Bowen Jin, Zhen Liu, Weiqi Wang, Bo Li, and Jieyu Zhao. Recut: Balancing reasoning length and accuracy in llms via stepwise trails and preference optimization. *arXiv preprint arXiv:2506.10822*, 2025.

[12] Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24312–24320, 2025.

[13] Joshua Ong Jun Leang, Aryo Pradipta Gema, and Shay B Cohen. Comat: Chain of mathematically annotated thought improves mathematical reasoning. *arXiv preprint arXiv:2410.10336*, 2024.

[14] Quentin Lhoest, Albert Villanova Del Moral, Yacine Jernite, Abhishek Thakur, Patrick Von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, et al. Datasets: A community library for natural language processing. *arXiv preprint arXiv:2109.02846*, 2021.

[15] Kaixuan Lou, Yifan Zhang, Yuanzhe Li, Zhiyuan Wu, Qianru Sun, Yikang Jiang, and Yisen Wang. Adacot: Pareto-optimal adaptive chain-of-thought triggering via reinforcement learning. *arXiv preprint arXiv:2505.11896*, 2025.

[16] Feng Luo, Yu-Neng Chuang, Guanchu Wang, Hoang Anh Duy Le, Shaochen Zhong, Hongyi Liu, Jiayi Yuan, Yang Sui, Vladimir Braverman, Vipin Chaudhary, et al. Autol2s: Auto long-short reasoning for efficient large language models. *arXiv preprint arXiv:2505.22662*, 2025.

[17] Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*, 2025.

[18] math ai. AIME 25 Dataset, 2025. URL `https://huggingface.co/datasets/math-ai/aime25`. Accessed: 2025-09-23.

[19] Maxwell-Jia. AIME2024, 2025. URL `https://huggingface.co/datasets/Maxwell-Jia/AIME_2024`. Accessed: 2025-09-23.

[20] Sania Nayab, Giulio Rossolini, Marco Simoni, Andrea Saracino, Giorgio Buttazzo, Nicolamaria Manes, and Fabrizio Giacomelli. Concise thoughts: Impact of output length on llm reasoning and cost. *arXiv preprint arXiv:2407.19825*, 2024.

[21] OpenAI. Learning to Reason with LLMs, 2024. URL `https://openai.com/index/learning-to-reason-with-llms/`. Accessed: 2025-09-19.

[22] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[23] Avinash Patil and Aryan Jadon. Advancing reasoning in large language models: Promising methods and approaches. *arXiv preprint arXiv:2502.03671*, 2025.

[24] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR, 2023.

[25] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

[26] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

[27] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[28] Guanchu Wang, Junhao Ran, Ruixiang Tang, Chia-Yuan Chang, Yu-Neng Chuang, Zirui Liu, Vladimir Braverman, Zhandong Liu, and Xia Hu. Assessing and enhancing large language models in rare disease question-answering. *arXiv preprint arXiv:2408.08422*, 2024.

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

[30] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

[31] Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.

[32] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[33] Minglai Yang, Ethan Huang, Liang Zhang, Mihai Surdeanu, William Wang, and Liangming Pan. How is llm reasoning distracted by irrelevant context? an analysis using a controlled benchmark. *arXiv preprint arXiv:2505.18761*, 2025.

[34] Junchi Yao, Shu Yang, Jianhua Xu, Lijie Hu, Mengdi Li, and Di Wang. Understanding the repeat curse in large language models from a feature perspective. *arXiv preprint arXiv:2504.14218*, 2025.

[35] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[36] Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adaptthink: Reasoning models can learn when to think. *arXiv preprint arXiv:2505.13417*, 2025.

[37] Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V Le, Ed Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. Self-discover: Large language models self-compose reasoning structures. *Advances in Neural Information Processing Systems*, 37: 126032–126058, 2024.