What Makes a Good Reasoning Chain? Uncovering Structural Patterns in Long Chain-of-Thought Reasoning

Gangwei Jiang[1,4]*, Yahui Liu[2]*, Zhaoyi Li[1,4], Victoria W.[2], Fuzheng Zhang[2], Linqi Song[4,5]†, Ying Wei[3]†, Defu Lian[1] †

[1]University of Science and Technology of China, [2]Kuaishou Technology, [3]Zhejiang University, [4]City University of Hong Kong, [5]City University of Hong Kong Shenzhen Research Institute

gwjiang@mail.ustc.edu.cn

Abstract

Recent advances in reasoning with large language models (LLMs) have popularized Long Chain-of-Thought (LCoT), a strategy that encourages deliberate and step-by-step reasoning before producing a final answer. While LCoTs have enabled expert-level performance in complex tasks, how the internal structures of their reasoning chains drive, or even predict, the correctness of final answers remains a critical yet underexplored question. In this work, we present LCoT2Tree, an automated framework that converts sequential LCoTs into hierarchical tree structures and thus enables deeper structural analysis of LLM reasoning. Using graph neural networks (GNNs), we reveal that structural patterns extracted by LCoT2Tree, including exploration, backtracking, and verification, serve as stronger predictors of final performance across a wide range of tasks and models. Leveraging an explainability technique, we further identify critical thought patterns such as over-branching that account for failures. Beyond diagnostic insights, the structural patterns by LCoT2Tree support practical applications, including improving Best-of-N decoding effectiveness. Overall, our results underscore the critical role of internal structures of reasoning chains, positioning LCoT2Tree as a powerful tool for diagnosing, interpreting, and improving reasoning in LLMs. The code is available at our GitHub repository.

1 Introduction

Large Language Models (LLMs) have achieved remarkable progress in nature language understanding and processing, with recent developments extending their capabilities to more complex reasoning tasks. Cutting-edge models such as OpenAI o3 (OpenAI, 2025) and DeepSeek R1 (Guo et al., 2025) push this frontier by emulating System 2 thinking (Li et al., 2025b), *i.e*, engaging in slow, deliberate, and step-by-step reasoning before arriving at a final answer. This approach, well known as Long Chain-of-thought (LCoT) reasoning (Chen et al., 2025; Gandhi et al., 2025), has empowered LLMs to achieve expert-level performance in challenging tasks such as mathematics, code generation, and scientific problem-solving (Seed et al., 2025; Team et al., 2025; Team, 2024). Despite their growing adoption, LCoTs remain largely a black box in one key aspect: *what makes a good thought chain?*

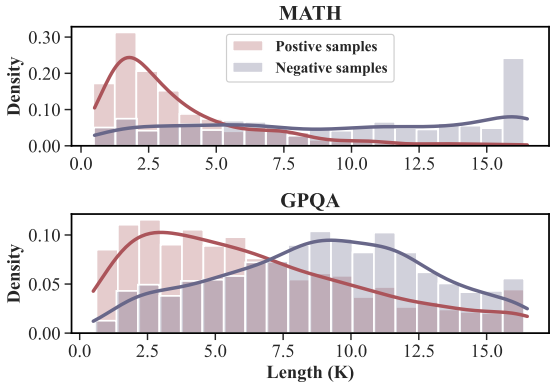

Figure 1: The distribution of output token length for correctly answered (Positive) and incorrectly answered (Negative) samples by DeepSeek-R1-Distill-Qwen-32B on two datasets.

Before the emergence of LCoT, researchers attempted to answer this question from a semantic perspective, often using process reward models (PRMs) that provide token-level or step-wise supervision based on logical coherence and factual accuracy (Xia et al., 2025; Zhang et al., 2025). While effective for short or moderately long CoTs, PRMs struggle to scale effectively as the length and structure complexity of reasoning chains increase (He et al., 2025). In the LCoT era, recent work has increasingly emphasized the importance of reasoning structure (Gandhi et al., 2025; Li et al., 2025a; Ye et al., 2025). Both Wu et al. and Ballon et al. highlighted the overthinking phenomenon,

* Equal contribution. This work was done when the author Gangwei Jiang was at Kuaishou Technology for intern.

† Corresponding author.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 6490–6514
November 4-9, 2025

where overly long reasoning chains can degrade rather than improve final answer quality. However, our analysis (Figure 1) shows that response length alone remains an inadequate predictor of answer correctness, as responses with similar lengths vary greatly in correctness. These findings suggest that these heuristics, such as token count, step count, or PRM-based semantic metrics fall short in effectively dictating reasoning success.

Thus, we propose Long Chain-of-Thought to Tree (LCoT2Tree), the first automated framework for structural analysis of reasoning in LLMs. LCoT2Tree transforms sequential LCoTs into hierarchical tree representations (Section 3.2), which enable structural patterns in reasoning chains, including exploration, backtracking, and verification, to be made explicit and analyzable. By modeling these trees with graph neural networks (GNNs), we not only extract these structural patterns as features, but also demonstrate that they serve as strong predictors of reasoning success (Section 3.3).

Beyond establishing their predictive power, we further investigate which structural patterns specifically contribute to reasoning success or failure, how these patterns vary across tasks and models, and how they can be applied to further enhance LLM reasoning in practice. Concretely, by leveraging a GNN-based explainability method, we unveil key thought patterns (*i.e.*, critical substructures within the tree) that explain answer correctness across diverse tasks and models (Section 4). These analyses reveal how reasoning behaviors differ by (1) answer correctness, (2) task type, and (3) model variant. Furthermore, we demonstrate that these patterns can be leveraged to improve Best-of-N decoding: incorporating our tree-based predictive classifier into its selection strategy consistently enhances accuracy across diverse models and tasks (Section 5). We summarize the main contributions of the proposed LCoT2Tree in three aspects:

- (*Predictability*) We are the first to explicitly construct structural representations of LCoT; our proposed LCoT2Tree offers stronger signals for reasoning success and improves binary classification of answer correctness by an average of **5.63%**, compared to using length alone.
- (*Interpretability*) We leverage LCoT2Tree to pinpoint the reasoning patterns that oftentimes lead to errors, *e.g.*, over-branching, and to account for disparate behaviors across tasks and models.
- (*Practicality*) We demonstrate that LCoT2Tree offers a principled path for selecting well-structured reasoning chains, greatly enhancing Best-of-N decoding and also remaining extensible for future decoding strategies.

2 Related Works

Reasoning LLMs Advancing reasoning capabilities of LLMs has shown benefits in tackling complex tasks (Kojima et al., 2022; Wei et al., 2022; Li et al., 2025b). Researchers first demonstrated that CoT prompting can significantly improve performance on complex tasks like arithmetic (Wei et al., 2022). To refine the reasoning processes, hierarchical cognitive phases have been introduced, such as multi-path exploration (Wang et al., 2023b; Yao et al., 2023), step verification (Miao et al., 2024; Gou et al., 2024), and iterative refinement (Madaan et al., 2023; Besta et al., 2024). These approaches expand solution spaces and deepen reasoning, driving more reliable answers. More recently, models such as Deepseek-R1 (Guo et al., 2025), Kimi-1.5 (Team et al., 2025) and QwQ-32B (Team, 2024) have leveraged rule-based reinforcement learning to embed reasoning capabilities directly into model parameters, achieving remarkable progress in handling complex tasks (Chen et al., 2025; Gandhi et al., 2025).

Chain-of-Thought Analysis Numerous studies have explored when CoT prompting is effective. Empirical research has revealed that factors, such as step length (Jin et al., 2024), relevance, the order of reasoning fragments (Wang et al., 2023a), and prompt structure (Li et al., 2025a), heavily influence performance. Expanding on these findings, Feng et al. (2023) and Chen et al. (2024a) proposed that there is an inherent reasoning limit in LLMs when tackling tasks exceeding a complexity threshold. In the context of long CoT, research has increasingly emphasized the importance of response structures in enhancing reasoning success (Gandhi et al., 2025; Muennighoff et al., 2025; Ye et al., 2025). Additionally, challenges like the overthinking phenomenon, where overly long responses inadvertently hurt the model performance, seemingly establish the correlation between length and reasoning success (Chen et al., 2024b; Wu et al., 2025; Cuadron et al., 2025). Beyond these prior works, we are motivated to develop an automated tool to empirically identify the structural patterns that dictate reasoning success in long CoT.

Besides the above analyses towards reasoning success, another line of works primarily analyzes

towards the semantic rationality of reasoning. Thus, early methods directly compare generated steps to human-annotated explanations (Welleck et al., 2022). However, such methods often fail to capture logical coherence beyond surface-level similarity. Recent work has introduced LLM-driven PRMs (Ling et al., 2023; Yuan et al., 2024; Zhang et al., 2025) to provide holistic and step-wise assessment, but they struggle with scaling to long and complex CoT reasoning chains (He et al., 2025). Rather than relying on surface similarity or token-level reward signals, we analyze reasoning success through internal structural patterns derived from hierarchical tree representations. The structural patterns offer a principled alternative for dictating "good" chains, and are fully complementary to this body of semantic-based research.

3 LCoT2Tree: Automated Long Chain-of-Thoughts to Tree

In this section, we empirically study overthinking, highlighting issues with assessing reasoning quality via CoT length. Then, we propose Long Chain-of-Thought to Tree (LCoT2Tree), an automated tool that converts LCoTs into tree structures to reveal cognitive frameworks and enable deeper analysis of LLMs' reasoning processes.

3.1 Overthinking Phenomenon

The "overthinking" phenomenon in reasoning models refers to situations where a model expends excessive computational resources (*e.g.*, generating overly long sequences or repeating reasoning steps), yet makes little contributions to the correctness of final answer. In some cases, this can even lead to a decline in performance (Chen et al., 2024b; Wu et al., 2025). Figure 2 illustrates this phenomenon by showing the relationship between the output token length and the answer accuracy of DeepSeek-32B (*i.e.*, DeepSeek-R1-Distill-Qwen-32B (Guo et al., 2025)) on the MATH (Hendrycks et al., 2021) dataset. It demonstrates that as the reasoning chain becomes unnecessarily long, model performance deteriorates, highlighting how overthinking can harm the reasoning ability of LLMs.

To tackle this issue, researchers have proposed using a length penalty during the training period to constrain the length of generated LCoTs (Team et al., 2025; Yu et al., 2025). However, this strategy relies on the oversimplified assumption that shorter or moderately long reasoning chains inherently lead to better reasoning quality. In this work, we conduct a classification experiment to empirically quantify the actual relationship between these two factors and uncover the limitations of relying on length as an indicator of reasoning quality.

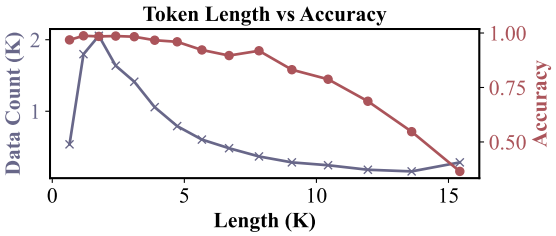

Figure 2: Data count and accuracy of the MATH dataset for DeepSeek-32B across varying response lengths. Accuracy notably declines as response length increases.

Experimental Setup. For our study, we use DeepSeek-32B, DeepSeek-R1 (Guo et al., 2025), QwQ-32B (Team, 2024), Seed-1.5-Thinking-pro (Seed et al., 2025), and Grok-3-mini-beta (xAI, 2025) as the primary models. We evaluate these models on four benchmark datasets: MATH (Level5 question in high school math competitions; Hendrycks et al., 2021), GPQA ("main" subset in grade-level google-proof question answering; Rein et al., 2024), LiveCodeBench (version 5 in live code benchmark; Jain et al., 2025), and MMLU-Pro (proficient-level multi-discipline language understanding; Wang et al., 2024). For each dataset, we collect 2,000 model responses, consisting of 1,000 correctly answered cases (Positive) and 1,000 incorrectly answered cases (Negative). These samples are divided into training and testing sets at a ratio of 4:1.

In our experiments, we train a logistic regression model using LCoT response length as the input feature and answer correctness as the target label. The test set accuracy quantifies the degree of correlation between LCoT length and reasoning quality-higher accuracy suggests a stronger association between these two factors.

Results and Analysis. Figure 1 shows that while longer outputs sometimes correlate with lower accuracy (e.g., in MATH), this trend does not generalize. In MMLU-Pro, substantial overlap exists between correct and incorrect responses across both moderate (5.0–10.0K) and very long (>15.0K) outputs, indicating that token length alone is an unreliable predictor of reasoning quality. Consistently, Table 1 reports only 60.0% and 58.0% accuracy for DeepSeek-32B and QwQ-32B, respectively, when using length-based classification. These relatively

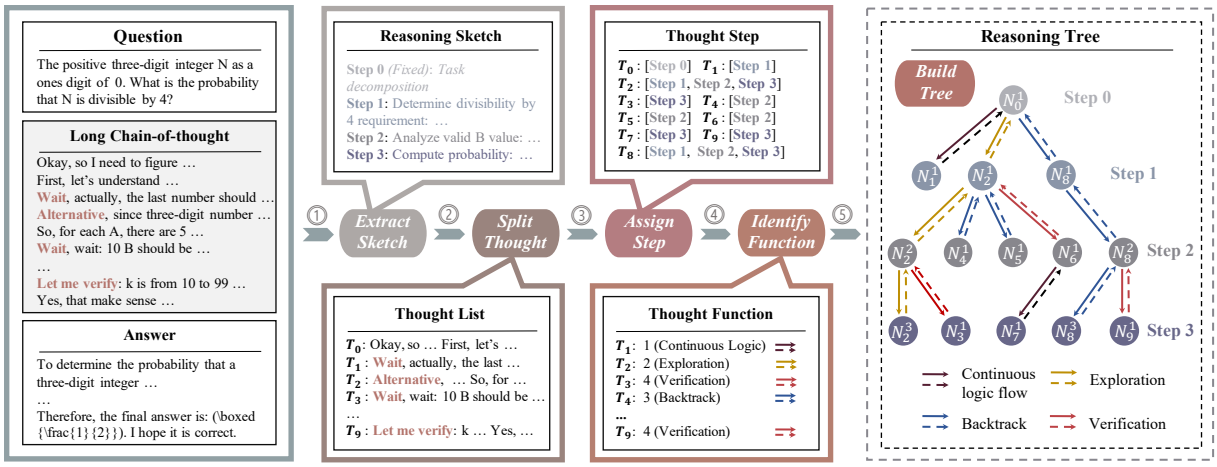

Figure 3: The workflow for LCoT2Tree. It transforms sequential long chain-of-thought into reasoning tree, which involves five steps: (1) Extract Sketch, (2) Split Thought, (3) Assign Step, (4) Identify Function, and (5) Build Tree.

low accuracies underscore the limitations of using response length alone in predicting reasoning success.

3.2 LCoT2Tree Tool

We present LCoT2Tree, a novel tool that extracts structural insights from LCoTs, addressing the limitations of length-based prediction. LCoT2Tree converts the sequential chain of reasoning into a tree structure, enabling a deeper analysis of cognitive behaviors such as exploration, backtracking, and reasoning depth. These components increasingly recognized as crucial for developing reasoning LLMs (Chen et al., 2025; Gandhi et al., 2025; Ye et al., 2025). To our knowledge, our work is the first to explicitly extract this structural information and conduct a quantitative analysis of its correlation with reasoning quality.

The LCoT2Tree tool involves five automated stages that transform a LCoT into an organized tree structure using an LLM (DeepSeek-v3; Liu et al., 2024a), as shown in Figure 3:

Stage 1: Extract Sketch. Leveraging the LLM with prompting (Figure 6), we condense the LCoT into a concise *Reasoning Sketch* that outlines its main reasoning steps. This sketch provides the **global structure** of the reasoning tree, with each extracted step serving as a logical anchor to guide the alignment of subsequent thoughts—for example, indicating which earlier step a backtracking thought refers to or whether any intermediate steps have been skipped.

Stage 2: Split Thought. Each "*Thought*" is defined as a consecutive segment in the reasoning chain without logical transitions (*e.g.*, exploration or verification within a step). In other words, a *Thought* serves as the basic unit of reasoning, enabling the analysis of local logical relations between adjacent units. We utilize common linguistic cues (*e.g.*, "Wait", "Alternatively", and "Let me verify") indicative of transitions between reasoning steps to segment the full reasoning chain into distinct fragments, yielding a *Thoughts List*.

Stage 3: Assign Step. Each thought in the *Thoughts List* is matched to one or more steps in the *Reasoning Sketch*, depending on its role in the overall reasoning process. This alignment is carried out using an LLM with prompting (Figure 7), generating a *Thought Step* dictionary that maps each thought to its corresponding reasoning depths.

Stage 4: Identify Function. By prompting the LLM (Figure 8), we analyze consecutive thought pairs to determine the later thought's role relative to the former, with possible roles: (1) Continuous Logic; (2) Exploration; (3) Backtracking; and (4) Verification. This assigns a *Thoughts Function* label to each thought for clearer reasoning-flow purpose understanding.

Stage 5: Build Tree. Finally, we organize the segmented thoughts into a hierarchical tree structure. Each node N_i^j in the tree corresponds to the i-th thought T_i, where j indicates how many times T_i has appeared. The placement of a node is determined by the *Thought Step*, and each edge represents a transition to a deeper level of reasoning, with the edge type defined by the *Thought Function* of its child node. When inserting a new thought T_i, we first identify the ordered list of reasoning steps it maps to, denoted as $[S_i^1, ..., S_i^n]$. Here, n indicates that the current thought encompasses n reasoning steps. Consequently, we create n nodes $N_i^1, ..., N_i^n$, where each node N_i^j represents the

		MATH	GPQA	LiveCodeBench	MMLU-Pro	4 Datasets	Average
DeepSeek-32B	Length-based	74.13%	67.08%	81.59%	59.95%	66.27%	69.80%
	Tree-based	80.81%	70.37%	82.21%	72.41%	71.14%	75.39%
	Gain	**+6.68%**	**+3.29%**	**+0.62%**	**+12.46%**	**+4.87%**	**+5.58%**
QwQ-32B	Length-based	75.82%	62.09%	78.30%	58.00%	66.97%	68.24%
	Tree-based	77.63%	68.55%	80.05%	72.58%	70.98%	73.96%
	Gain	**+1.81%**	**+6.46%**	**+1.75%**	**+14.58%**	**+4.01%**	**+5.72%**
DeepSeek-R1	Length-based	76.94%	69.57%	81.75%	63.00%	70.54%	72.36%
	Tree-based	80.30%	73.56%	81.80%	75.85%	73.72%	77.05%
	Gain	**+3.36%**	**+3.99%**	**+0.05%**	**+12.85%**	**+3.18%**	**+4.69%**
Seed-1.5-Thinking-pro	Length-based	67.48%	64.84%	76.39%	63.59%	64.34%	67.33%
	Tree-based	70.07%	69.81%	77.72%	70.82%	67.68%	71.22%
	Gain	**+2.59%**	**+4.97%**	**+1.33%**	**+7.23%**	**+3.34%**	**+3.89%**
Grok-3-mini-beta	Length-based	71.31%	61.48%	84.77%	55.47%	63.87%	67.38%
	Tree-based	83.18%	66.79%	86.35%	70.68%	71.26%	75.65%
	Gain	**+11.87%**	**+5.31%**	**+1.58%**	**+15.21%**	**+7.40%**	**+8.27%**

Table 1: Comparison of performance across various reasoning LLMs and datasets using the length-based method and our proposed tree-based approach for classifying response correctness based on LCoT information. Classification results are reported as the average over five runs.

portion of the thought aligned with the S_i^j-th step. The insertion process follows two rules: (1) If S_i^1 is greater than the step of the latest node N_{i-1}^j in the tree, the new node N_i^1 is added as a child of N_{i-1}^j. (2) Otherwise, we backtrack to the most recent node at step $S_i^1 - 1$. Then we create a new branch from that node and link it to new node N_i^1.

Once N_i^1 is determined, the remaining nodes $N_i^2, ..., N_i^n$ are added sequentially and connected to the previous one. For example, in Figure 3, when inserting T_8 to the tree, its associated reasoning steps $[S_8^1, S_8^2, S_8^3] = [1, 2, 3]$, as determined by the *Thought Step*. At that point, the latest node in tree is N_7^1, which is at step 3 (greater than 1). Therefore, we backtrack to the latest node at step 0, N_0^1, and attach N_8^1 as its child. After that, N_8^2 and N_8^3 are linked sequentially to N_8^1 and N_8^2, respectively.

In the end, we extract the tree structure, showing how thoughts are connected and branch throughout the reasoning process. This structural representation offers three key benefits: (1) highlights key cognitive patterns (*e.g.*, exploration, backtracking and verification); (2) supports more accurate assessment of reasoning quality; and (3) enables structure-aware analysis of reasoning behaviors. Implementation details, including prompts and case visualizations, are available in Appendix A.

3.3 Effectiveness of LCoT2Tree

To assess the effectiveness of the LCoT2Tree tool, we conduct a quantitative evaluation by using graph neural networks (GNNs) to predict answer correctness based on the tree structures extracted from LCoTs. This evaluation demonstrates the practical value of tree-based representations for understanding complex reasoning processes.

Experimental Setup. We use the same dataset described in Section 3.1, which contains responses from five reasoning models across four public benchmarks. The key difference is that we extract the tree structure from each LCoT response and use it as input to the GNNs. Our objective is to assess how effectively these tree structures can distinguish between correct and flawed reasoning. To this end, we utilize GATv2 (Brody et al., 2022), a GNN architecture suited for modeling hierarchical structures and their relationships. The model takes the nodes, edges, and associated features of each LCoT tree as input and learns a structural embedding that represents the overall reasoning pattern. Implementation details and graph construction are provided in Appendix B. We use classification accuracy as the evaluation metric. A high accuracy score indicates that the model successfully captures the correlation between reasoning structure and answer correctness.

Effectiveness across Tasks. Table 1 shows the classification results using tree-based input, compared to baseline methods that rely on the length-based feature. We assess how well the tree-based method generalizes across diverse types of reasoning tasks, including MATH, GPQA, LiveCodeBench, MMLU-Pro, and a combined dataset of these benchmarks. Across all tasks, the tree-based method consistently outperforms the length-based baseline. The improvement is particularly

notable on MMLU-Pro, a dataset where reasoning correctness is difficult to predict from token length alone. For example, our method achieves substantial accuracy gains of +12.46% and +14.58% on DeepSeek-32B and QwQ-32B, respectively. Even on datasets like LiveCodeBench, where the length-based approach already performs strongly, the tree-based method still yields improvements, demonstrating its robustness.

Effectiveness across Models. For the generalizability of our method, the tree-based classifier consistently achieves higher accuracy than the length-based baseline across all models. Average accuracy gains range from +3.89% (Seed-1.5-Thinking-pro) to +8.27% (Grok-3-mini-beta), indicating that the LCoT2Tree provides more informative and reliable structural representations of reasoning processes.

These quantitative evaluation validates the effectiveness of the LCoT2Tree tool across diverse tasks and models. By capturing deeper structural and cognitive patterns in reasoning, it enables more accurate prediction of reasoning success.

3.4 Coverage of LCoT2Tree on Reasoning Structure

An important question is how well tree-based representations capture the diversity of reasoning behaviors. In LCoT2Tree, reasoning is modeled as a hierarchical tree with typed edges, covering four core categories: *Continuous Logic*, *Exploration*, *Backtracking*, and *Validation* (see Appendix A for details). This taxonomy provides broad coverage of reasoning dynamics observed in long chains of thought. For example, exploratory branches naturally represent alternative problem-solving paths, while loop-like behaviors such as self-correction are treated as instances of *Backtracking*, where the model revisits earlier steps and generates revised conclusions. Although the tree is inherently acyclic, typed edges enable the representation of loop-like relations and functional dependencies among sibling branches.

The primary limitation of tree structures lies in their inability to fully capture cases where a thought simultaneously integrates multiple parent contexts. This suggests a natural extension toward graph-based representations, which could more faithfully model such interactions. Nonetheless, for the predominant reasoning patterns—exploration, backtracking, and validation—the tree abstraction provides a clear, interpretable, and computationally tractable foundation for structural analysis. Overall, LCoT2Tree shows strong potential as an automated tool for analyzing, evaluating, and improving the behavior of reasoning systems.

4 Understand Behaviors of Reasoning Large Language Models

In this section, we leverage LCoT2Tree to analyze and understand reasoning behaviors. First, we identify key thought patterns in the reasoning tree that predict errors. Then, we compare behaviors across tasks and models. Our findings show that reasoning varies by (1) output correctness, (2) task type, and (3) model variant, underscoring the importance of structural information in reasoning analysis.

Explainability Method. To interpret the model's predictions on reasoning quality and uncover the influential reasoning patterns, we adapt a graph explainability method called *GNNExplainer* (Ying et al., 2019). This method uncovers important subgraphs by maximizing the mutual information between the GNN's output and the distribution of possible subgraph structures. These extracted subgraphs also correspond to critical thought patterns within the reasoning chain. For example, in models trained to predict incorrect answers, the highlighted subgraphs often reflect flawed reasoning behaviors that lead to poor performance. Similarly, in models trained on MATH tasks, the important subgraphs typically capture common reasoning patterns observed in mathematical problem-solving.

4.1 Error Patterns in LCoT

The experiments in Section 3.3 suggest that reasoning trees of model responses exhibit separable structures for correct and incorrect outcomes. To further explore the behaviors that contribute to failures, we employ GNNExplainer to identify the most influential edges in each reasoning tree. This allows us to extract critical subgraphs from incorrect responses and summarize common patterns across diverse examples (See details in Appendix D.1). The identified error patterns are visualized in the top portion of Figure 4, with detailed examples shown in Figure 9. Additionally, we analyze 100 error responses from three tasks and report the frequency of each pattern in the bottom portion of Figure 4. A key observation is that excessive and insufficient branching are both strongly associated with incorrect reasoning.

	Task-specific Analysis				Model-specific Analysis	
	MATH/GPQA	MATH/LCB	MATH/MMLU	GPQA/LCB	DS-32/DS-R1	DS-32/Grok
Length-based	50.45%	63.72%	69.43%	60.65%	55.17%	61.06%
Tree-based	83.51%	89.22%	78.46%	85.55%	67.88%	93.22%
Gain	**+33.06%**	**+25.50%**	**+9.03%**	**+24.90%**	**+12.71%**	**+32.16%**

Table 2: Comparison of task-specific and model-specific classification accuracy using the length-based method and the proposed tree-based approach. Task-specific analysis is conducted on the DeepSeek-32B model across different datasets, while model-specific analysis is performed on the MATH dataset across multiple model variants.

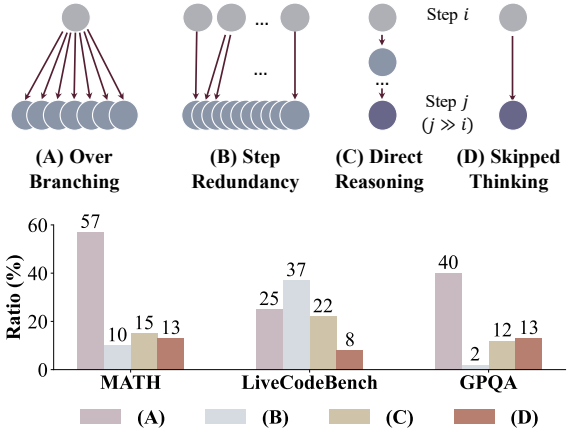

Figure 4: Visualization and frequency of four structural error patterns across three datasets. (A) *Over Branching*: abundance of explorations or verifications within a single node, (B) *Step Redundancy*: over-generation of thoughts within a single reasoning step, (C) *Direct Reasoning*: following a straight, minimal-branch path from one step to a much deeper step, and (D) *Skipped Thinking*: jumping multiple steps ahead without intermediate logical analysis. Representative example of each class is available in Figure 9.

4.2 Task-Specific Patterns in LCoT

In the left part of Table 2, we present the results of a task separability experiment conducted on the DeepSeek-32B model. We classify reasoning trees across task pairs (*e.g.*, MATH/GPQA, MATH/LCB, MATH/MMLU-Pro, GPQA/LCB), with additional results for QwQ-32B provided in Appendix C.1. The dataset follows the same construction as in Section 3.1, but labels tasks instead of correctness. Results show that our tree-based method effectively distinguishes task-specific reasoning patterns, achieving an average accuracy of 84.19%. Notably, in cases where length-based features fall short, such as MATH/GPQA and GPQA/LCB, tree-based representations yield substantial gains of +33.06% and +24.90%, respectively, highlighting their strength in capturing deeper reasoning patterns.

Discovering Task-Specific Reasoning Patterns. Beyond quantitative separation, we further leverage LCoT2Tree to reveal task-specific reasoning patterns through qualitative analysis. Main conclusions, based on DeepSeek-32B, are as follows: For MATH (Figure 10), the reasoning trees exhibit a diagonally descending structure, reaching deeper steps through repeated backtracking. This reflects a layered, step-by-step problem-solving approach. In contrast, the behaviors in code completion (Figure 11) show wide, parallel branches with minimal exploration or verification, reflecting a more straightforward pattern of generation. GPQA (Figure 12) samples reveal high out-degree nodes, where the model repeatedly revisits complex concepts, indicating the model's uncertainty in dealing an expert-level question. Meanwhile, the trees of MMLU-Pro (Figure 13) are relatively shallow with minimal branching, reflecting a straightforward deductive reasoning style that aligns with the nature of knowledge-based questions. These observations highlight LCoT2Tree's ability to provide interpretable insights into the distinct reasoning strategies employed across different task types. Detailed case studies are provided in Appendix D.2 with visualizations shown in Figure 10 - Figure 13.

4.3 Model-Specific Patterns in LCoT

We explore whether different models exhibit distinguishable reasoning behaviors on the same dataset. The results, shown in the right part of Table 2, demonstrate that LCoT2Tree effectively captures model-specific patterns. In particular, tree-based representations significantly outperform simple length-based features, with gains of +12.71% for DS-32 (DeepSeek-32B) vs. DS-R1 (DeepSeek-R1), and +32.16% for DS-32 vs. Grok (Grok-3-mini-beta). Notably, the relatively lower separability score between DS-32 and DS-R1 (67.88%) can be attributed to the fact that DS-32 is a distilled version of DS-R1. In contrast, DS-32 and Grok show a high separability of 93.22%, suggesting fundamentally different reasoning styles driven by architectural and training differences. Addi-

tional results (Appendix C.2) show that QwQ-32B aligns more closely with the DeepSeek family than with Grok or Seed (Seed-1.5-Thinking-pro). These findings again highlight the strength of structural representations in revealing fine-grained behavioral distinctions.

Discovering Model-Specific Reasoning Patterns. To complement the quantitative analysis, we also conduct a qualitative comparison of reasoning trees across different models on the MATH dataset (Appendix D.3). Our analysis reveals that both DS-R1 (Figure 14) and QwQ-32B (Figure 15) produce reasoning structures similar to DS-32 (Figure 10), consistent with quantitative results. However, DS-R1 tends to prune its reasoning paths earlier, suggesting a more aggressive backtracking strategy. In contrast, QwQ-32B shows more extensive exploration in the later stages of reasoning. On the other hand, Seed (Figure 16) and Grok (Figure 17) follow simpler, more linear reasoning paths with fewer thought transitions and minimal branching, reflecting a straightforward reasoning strategy.

4.4 Shortcomings in Understanding LCoT from the Structural Perspective

Correct Structure but Wrong Output. Despite structurally valid reasoning paths, models can still produce incorrect answers due to semantic errors like misinterpreting the problem, making calculation mistakes, or failing conditional logic. This indicates that reasoning LLMs do not consistently exhibit behaviors like backtracking or verification when facing ambiguity or errors. These cases expose the limitation of structural analysis alone and suggests that combining structural insights with semantic verification is necessary for comprehensive reasoning understanding.

Flawed Structure but Correct Output. Using our classifier, we identify a set of responses with correct final answer but weak or flawed reasoning. These cases often involve reasoning paths that deviate from systematic problem-solving, including guessing, brute-force enumeration, or overly late-stage corrections. Such cases underscore the limitations of using answer correctness alone to assess reasoning quality, as it tolerates shallow or unsound reasoning paths. Addressing these flawed-but-correct patterns can guide LLMs toward producing reasoning that is not only accurate but also logically sound.

5 Application of LCoT2Tree: Tree-based Best-of-N Decoding

Beyond evaluating the quality of model reasoning (Section 3.3), we put forward practical application to support the decoding process in LLMs. Specifically, we propose an approach to improve the reasoning quality during the decoding stage by selecting the best model response from multiple candidates with the tree-based classifier.

Method. Best-of-N decoding is a widely used strategy for improving the quality of responses generated by LLMs (Wu et al., 2024; Snell et al., 2024; Brown et al., 2024). In this strategy, the model produces N candidate outputs, and a final response is selected based on a scoring function. However, conventional scoring methods, based on surface-level heuristics or reward models, often ignore the impact of output structures. This limitation can lead to suboptimal choices, especially in tasks that require deep or structured reasoning.

To this end, we incorporate LCoT2Tree into the Best-of-N decoding framework to guide the selection of high-quality reasoning outputs. Our method involves three main steps: (1) For each candidate response, we use LCoT2Tree to build its corresponding reasoning tree; (2) A graph-based classifier, trained to distinguish between successful and flawed reasoning structures, assigns a score to each candidate based on its structural features; (3) The candidate with the highest score is chosen as the final output.

Experiments. We choose LiveCodeBench (LCB) as our primary benchmark. We train the GNN models following the setup in Section 3.3 using LCB-v5 dataset and then evaluate on a challenging subset (filtered by correctness ratio) of the LCB-v6 dataset. We compare our tree-based Best-of-N decoding method with three baselines: (1) *ORM-Best* (Brown et al., 2024), which selects the response with the highest score from an outcome reward model (we use Skywork-Reward-Gemma-2-27B-v0.2 (Liu et al., 2024b)); (2) *PRM-Best*, which scores responses based on the product of step-level scores from a process reward model (*i.e.*, Qwen2.5-Math-PRM-72B (Zhang et al., 2025)); and (3) *Length-Best* (Wang et al., 2025), which selects the response with the fewest tokens. All experiments use $N = 10$ candidate responses. Additional results on MATH with more baselines are presented in Appendix C.3.

Results. As shown in Figure 5, our tree-based Best-

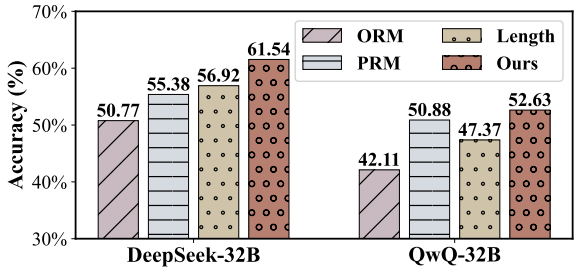

Figure 5: Accuracy comparison of different Best-of-N decoding strategies on the LCB-v6 benchmark.

of-N method outperforms both Length-Best, ORM-Best and PRM-Best on the LCB-v6 benchmark. For DeepSeek-32B, it achieves 61.54% accuracy, exceeding Length-Best by +4.62%, ORM-Best by +10.77% and PRM-Best by +6.16%. QwQ-32B shows similar gains, with our method reaching 52.63%, outperforming the baselines by +5.26%, +10.52% and 1.75%, respectively. These results highlight the advantage of using structural reasoning signals via LCoT2Tree to improve candidate selection in complex tasks.

6 Conclusion

In this work, we introduce a novel framework, named LCoT2Tree, for converting LCoT responses into hierarchical tree structures. LCoT2Tree enables more interpretable and structural analysis of complex reasoning processes, with significantly improving the prediction of reasoning success across a wide range of tasks and models. Beyond evaluation, we apply LCoT2Tree for behavioral analysis, revealing error patterns and accounting for disparate behaviors across tasks and models. Furthermore, we extend LCoT2Tree to a practical application by integrating it into the Best-of-N decoding paradigm, leading to more accurate outputs than ORM, PRM and length-based baselines. Collectively, these findings underscore the significance of structural reasoning analysis and establish LCoT2Tree as a promising tool for understanding and improving LLMs reasoning capabilities.

Limitations

While LCoT2Tree is a powerful framework for analyzing reasoning structures, several limitations remain. First, as discussed in Section 4.4, structural analysis alone cannot capture semantic errors or recognize correct reasoning that deviates from common structural patterns. To address this limitation, future work should integrate semantic reasoning signals with structural analysis to achieve a more holistic understanding of LLM reasoning behaviors.

Second, the effectiveness of structural analysis relies in part on the fact that current LLMs often generate reasoning that is incomplete or loosely organized. As models improve and begin to produce more coherent and well-structured reasoning by default, the value of structural diagnostics may decrease. Nevertheless, until such consistency is achieved, structural cues remain a valuable tool for identifying and improving reasoning quality.

Finally, the construction of reasoning trees in LCoT2Tree currently depend on *off-the-shelf* large language models (e.g., DeepSeek-V3 (Liu et al., 2024a)), which makes the pipeline computationally expensive.

Acknowledgments

The work was supported in part by the National Natural Science Foundation of China (No.U24A20253, No.62441236, and No.62371411), the Research Grants Council of the Hong Kong SAR under Grant Collaborative Research Fund C1042-23GF and GRF 11217823, InnoHK initiative, the Government of the HKSAR, Laboratory for AI-Powered Financial Technologies.

References

Marthe Ballon, Andres Algaba, and Vincent Ginis. 2025. The relationship between reasoning and performance in large language models–o3 (mini) thinks harder, not longer. *arXiv preprint arXiv:2502.15631*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Shaked Brody, Uri Alon, and Eran Yahav. 2022. How attentive are graph attention networks? In *International Conference on Learning Representations (ICLR)*.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wangxiang Che. 2025. Towards

reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Qiguang Chen, Libo Qin, Jiaqi Wang, Jingxuan Zhou, and Wanxiang Che. 2024a. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. *Advances in Neural Information Processing Systems (NeurIPS)*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024b. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, and 1 others. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2023. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems (NeurIPS)*.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. 2025. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations (ICLR)*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Yancheng He, Shilong Li, Jiaheng Liu, Weixun Wang, Xingyuan Bu, Ge Zhang, Zhongyuan Peng, Zhaoxiang Zhang, Zhicheng Zheng, Wenbo Su, and 1 others. 2025. Can large language models detect errors in long chain-of-thought reasoning? *arXiv preprint arXiv:2502.19361*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *The Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations (ICLR)*.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. In *Findings of the Association for Computational Linguistics ACL*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems (NeurIPS)*.

Dacheng Li, Shiyi Cao, Tyler Griggs, Shu Liu, Xiangxi Mo, Eric Tang, Sumanth Hegde, Kourosh Hakhamaneshi, Shishir G Patil, Matei Zaharia, and 1 others. 2025a. Llms can easily learn to reason from demonstrations structure, not content, is what matters! *arXiv preprint arXiv:2502.07374*.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, and 1 others. 2025b. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. *Advances in Neural Information Processing Systems (NeurIPS)*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024b. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems (NeurIPS)*.

Ning Miao, Yee Whye Teh, and Tom Rainforth. 2024. Selfcheck: Using LLMs to zero-shot check their own

step-by-step reasoning. In *International Conference on Learning Representations (ICLR)*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

OpenAI. 2025. OpenAI o3-mini.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *Conference on Language Modeling*.

ByteDance Seed, Yufeng Yuan, Yu Yue, Mingxuan Wang, Xiaochen Zuo, Jiaze Chen, Lin Yan, Wenyuan Xu, Chi Zhang, Xin Liu, and 1 others. 2025. Seed-thinking-v1. 5: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Qwen Team. 2024. Qwq: Reflect deeply on the boundaries of the unknown. *URL https://qwenlm. github. io/blog/qwq-32b-preview*.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023a. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, and 1 others. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems (NeurIPS)*.

Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. 2022. Naturalprover: Grounded mathematical proof generation with language models. *Advances in Neural Information Processing Systems (NeurIPS)*.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Scaling inference computation: Compute-optimal inference for problem-solving with language models. In *The Workshop on Mathematical Reasoning and AI at NeurIPS*.

Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*.

xAI. 2025. Grok 3 Beta — The Age of Reasoning Agents.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. 2025. Evaluating mathematical reasoning beyond accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems (NeurIPS)*.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. 2024. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

A LCoT2Tree Tool Implementation Details

The LCoT2Tree process involves five automated stages to transform a LCoT into an organized tree structure using the LLM (DeepSeek-v3; Liu et al., 2024a), as shown in Figure 3. Here, we introduce the detailed implementation of each step:

Stage 1: Extract Sketch. Leveraging the LLM with prompt 6, we condense the LCoT into a sketch that captures its core reasoning steps. This *Reasoning Sketch* provides an abstract of the reasoning process, focusing on the key steps and the logical flow of the reasoning.

Stage 2: Split Thought. In this stage, the LCoT is split into a list of thoughts. We first define a "*Thought*" as: a continue segment in a reasoning chain that involves no logical transition, such as exploration or verification. We then analyze the collected LCoTs to identify common linguistic patterns (*i.e.*, separators) that signal shifts between distinct reasoning steps. The separators set to ["Alternatively", "Hmm", "Let me verify", "let's verify", "To verify", "Wait", "Verify"] for Deepseek-32B, QwQ-32B, and Deepseek-R1. And extend with ["Let's confirm", "Let's check", "Another example", "But let's", "wait", "No:", "no:", "Now"] for Seed-1.5-thinking-pro and Grok-3-mini-beta. According to these markers, the long reasoning chain is divided into individual thoughts, forming a *Thoughts List* where each item represents a single reasoning fragment.

Stage 3: Assign Step. Each thought in the *Thoughts List* is then aligned with one or more corresponding steps in the *Reasoning Sketch*, based on its role in the overall reasoning process. This mapping is performed using an LLM with prompt 7, generating a *Thought Step* dictionary that captures the contextual meaning and reasoning stage (*i.e.*, depth) associated with each thought. To improve token efficiency, we group and merge adjacent thoughts before feeding them into the LLM. However, due to the large number of thoughts in the *Thought List*, processing them all at once is infeasible. Therefore, we segment the list into smaller batches, each containing consecutive thoughts with a combined word count of no more than 600. These batches are then input to the LLM, which returns the corresponding reasoning step for each thought in a single response.

Stage 4: Identify Function. We further analyze each pair of consecutive thoughts using LLM with prompt 8 to determine the role of the latter thought in relation to the former (*e.g.*, continuation, exploration, or verification). This step provides a more precise understanding of the relationships between individual thoughts within the reasoning process. Specifically, the roles are categorized as follows: (1) Continuous Logic – A direct continuation or extension of the reasoning in the previous thought. (2) Exploration – Introduces alternative reasoning paths, unrelated concepts, or new topics. (3) Backtracking – Revises, corrects, or adjusts the reasoning from the previous step. (4) Validation – Provides supporting evidence, justification, or examples for the previous thought. If the *Thought List* contains N thoughts, we perform $N-1$ LLM calls to analyze each adjacent pair.

Stage 5: Build Tree. Finally, we organize the segmented thoughts into a hierarchical tree structure. Each node N_i^j in the tree corresponds to the i-th thought T_i, where j indicates how many times T_i has appeared. The placement of a node is determined by the *Thought Step*, and each edge represents a transition to a deeper level of reasoning, with the edge type defined by the *Thought Function* of its child node. When inserting a new thought T_i, we first identify the ordered list of reasoning steps it maps to, denoted as $[S_i^1, ..., S_i^n]$. Here, n indicates that the current thought encompasses n reasoning steps. Consequently, we create n nodes $N_i^1, ..., N_i^n$, where each node N_i^j represents the portion of the thought aligned with the S_i^j-th step. The insertion process follows two rules: (1) If S_i^1 is greater than the step of the latest node N_{i-1}^j in the tree, the new node N_i^1 is added as a child of N_{i-1}^j. (2) Otherwise, we backtrack to the most recent node at step $S_i^1 - 1$. Then we create a new branch from that node and link it to new node N_i^1.

Once N_i^1 is placed, the remaining nodes $N_i^2, ..., N_i^n$ are added sequentially and connected to the previous one. For example, in Figure 3, when inserting T_8 to the tree, its associated reasoning steps $[S_8^1, S_8^2, S_8^3] = [1, 2, 3]$, as determined by the *Thought Step*. At that point, the latest node in tree is N_7^1, which is at step 3—greater than 1. Therefore, we backtrack to the latest node at step 0, N_0^1, and attach N_8^1 as its child. After that, N_8^2 and N_8^3 are linked sequentially to N_8^1 and N_8^2, respectively.

In the end, we successfully extract the whole tree structure using LCoT2Tree. To support interpretation, we provide a visualization tool for the generated reasoning tree, which allows users to in-

teractively explore the thought process behind each node while viewing the overall tree structure. Example screenshots of the visualization results are shown in Figures 10–17. In each figure, the right column displays the key reasoning steps identified in Step 1, and each node represents an individual thought. The solid line denotes the edge from father to child and the dash line denotes the edge from child to father. Edges are colored according to its function in reasoning process.

B Classification Implementation Detail

B.1 Dataset Construction

We use the same dataset as described in Section 3.1, consisting of response samples generated by five reasoning LLMs (LLMs) across four public benchmarks: MATH, GPQA, LiveCodeBench (LCB), and MMLU-Pro. Each sample is labeled as positive or negative with answer correctness, which serves as the ground truth for our binary classification task. To ensure sufficient data volume, we apply repeated sampling for each benchmark, generating up to 2,000 samples per dataset. For instance, the LCB benchmark contains 167 unique problems. By generating 16 responses per problem, we obtain approximately 1,000 correctly answered samples and 1,000 incorrect ones.

B.2 Graph Construction

For each response, we begin by applying the LCoT2Tree framework to convert it into a structured reasoning tree. Each node N_i^j in the tree corresponds to the i-th thought T_i, where j indicates how many times T_i has appeared. The placement of a node is determined by the *Thought Step*, and each edge represents a transition to a deeper level of reasoning, with the edge type defined by the *Thought Function* of its child node as introduced in 3.2. We then transform the tree into a graph representation. Notably, we construct bidirectional edges, allowing information to flow both from parent to child and from child to parent. This design enables the model to simulate behaviors like backtracking, which are often essential in complex reasoning. In the end, each sample produces a single graph instance for classification.

B.3 Node and Edge Features

We design informative features for both nodes and edges to enhance the performance of our tree-based classification model. For each node in the reasoning tree, we extract the following features: (1) the index of the current thought, (2) the reasoning depth of the current node, (3) the cumulative number of tokens used up to current node, (4) the number of child nodes, and (5) the cumulative number of nodes at the same reasoning depth.

For edge features, we assign each parent-to-child edge a feature based on its logical role as identified by LCoT2Tree: "1" for continuation, "2" for exploration, "3" for backtracking, and "4" for validation. To distinguish child-to-parent edges (used to capture reverse information flow, such as backtracking), we assign the same value but multiply it by -1. This setup helps the model differentiate directional semantics during message passing.

B.4 Hyperparameters

We adopt the GATv2 architecture (Brody et al., 2022) to model reasoning trees, leveraging its dynamic attention mechanism and improved capability for capturing hierarchical dependencies. The model comprises two GATv2 layers, each with a hidden size of 64. After message passing, graph-level embeddings are obtained via global mean pooling. These embeddings are then fed into a two-layer MLP with ReLU activation, serving as the classification head to predict whether a given reasoning structure leads to a correct or incorrect answer.

To train the model, we use binary cross-entropy loss and the Adam optimizer with a learning rate of 1e-3. The model is trained for up to 100 epochs with a batch size of 32. We split the training dataset into 90% for training and 10% for validation. All experiments are conducted using the PyTorch Geometric framework.

C Additional Experimental Results

C.1 Additional Results on Task-specific Analysis

In Table 3, we provide additional results from the task separability experiments using the DeepSeek-32B and QwQ-32B models. We classify reasoning trees across all task pairs, including MATH/GPQA, MATH/LCB, MATH/MMLU-Pro, GPQA/LCB, GPQA/MMLU-Pro, and LCB/MMLU-Pro. The findings are consistent with the conclusions presented in Section 4.2.

		MATH/GPQA	MATH/LCB	MATH/MMLU	GPQA/LCB	GPQA/MMLU	MMLU/LCB
DS-32	Length	50.45%	63.72%	69.43%	60.65%	77.71%	82.34%
	Tree	83.51%	89.22%	78.46%	85.55%	82.89%	92.12%
	Gain	**+33.06%**	**+25.50%**	**+9.03%**	**+24.90%**	**+5.18%**	**+9.78%**
QwQ-32	Length	52.51%	56.64%	67.38%	61.22%	68.25%	73.03%
	Tree	77.82%	67.88%	80.85%	85.69%	76.70%	87.20%
	Gain	**+25.31%**	**+11.24%**	**+12.17%**	**+24.47%**	**+8.45%**	**+14.17%**

Table 3: Comparison of task-specific classification accuracy using the baseline length-based method and the proposed tree-based representation.

	MATH	GPQA	LiveCodeBench	MMLU-Pro	4 Datasets
DeepSeek-32B	±0.0048	± 0.0037	±0.0024	± 0.0089	± 0.0043
QwQ-32B	±0.0023	±0.0025	±0.0030	±0.0079	± 0.0039
DeepSeek-R1	±0.0076	±0.0029	±0.0010	±0.0051	± 0.0018
Seed-1.5-Thinking-pro	±0.0037	±0.0061	±0.0041	±0.0066	± 0.0026
Grok-3-mini-beta	±0.0025	±0.0020	±0.0037	±0.0153	± 0.0041

Table 4: Standard deviation of our proposed tree-based approach on classifying response correctness based on LCoT information corresponding to Table 1.

		DS-32/DS-R1	DS-32/QwQ-32	DS-32/Seed	DS-32/Grok	DS-R1/Seed
MATH	Length-based	55.17%	61.49%	55.58%	61.06%	56.90%
	Tree-based	67.88%	70.93%	82.15%	93.22%	80.10%
	Gain	**+12.71%**	**+9.44%**	**+26.57%**	**+32.16%**	**+23.20%**
GPQA	Length-based	50.87%	51.12%	67.96%	49.43%	65.34%
	Tree-based	75.34%	61.60%	95.20%	99.42%	84.68%
	Gain	**+24.47%**	**+10.48%**	**+27.24%**	**+49.99%**	**+19.34%**
LCB	Length-based	54.49%	54.17%	52.37%	54.49%	53.39%
	Tree-based	86.32%	71.73%	96.12%	86.32%	82.51%
	Gain	**+31.83%**	**+17.56%**	**+43.75%**	**+31.83%**	**+29.12%**
MMLU	Length-based	55.36%	60.10%	54.17%	53.23%	59.55%
	Tree-based	62.86%	64.99%	73.65%	85.62%	71.89%
	Gain	**+7.50%**	**+4.89%**	**+19.48%**	**+32.39%**	**+12.34%**

Table 5: Comparison of model-specific classification accuracy using the baseline length-based method and the proposed tree-based representation.

	DeepSeek-32B		QwQ-32B	
	LiveCodeBench	MATH	LiveCodeBench	MATH
Vote	-	80.41%	-	**71.19%**
Length-Best	56.92%	56.70%	47.37%	55.93%
Length-Vote	-	67.01%	-	57.63%
ORM-Best	50.77%	60.82%	42.11%	57.63%
ORM-Vote	-	68.04%	-	67.80%
PRM-Best	62.89%	63.92%	50.88%	57.63%
PRM-Vote	-	62.89%	-	55.93%
Ours-Best	**61.54%**	65.98%	**52.63%**	67.80%
Ours-Vote	-	**82.47%**	-	**71.19%**

Table 6: Accuracy comparison of different Best-of-N decoding strategies on the two benchmark.

C.2 Additional Results on Model-specific Analysis

Table 5 presents the detailed analysis of whether different models display distinguishable reasoning behaviors when applied to the same dataset. The results confirm that LCoT2Tree effectively captures model-specific reasoning patterns that generalize across tasks. Specifically, QwQ-32B exhibits reasoning behaviors more closely aligned with the DeepSeek family, compared to Grok-3-mini-beta and Seed-1.5-Thinking-pro. These findings further underscore the effectiveness of structural representations in revealing subtle differences in model behavior.

C.3 Additional Results on Best-of-N Decoding

Table 6 provides a detailed comparison of different Best-of-N decoding strategies on the MATH and LiveCodeBench (LCB) datasets using responses from two LLMs: DeepSeek-32B (DS-32) and Qwen-32B (QwQ-32). For the MATH benchmark, we evaluate on samples from the MATH500 and Level5 subsets that are not included in the training set. For LCB, we use LCB-v6 as the test set. In both cases, we ensure that the selected test samples are challenging—each sample is incorrectly answered at least twice across 10 runs. We set $N = 10$ and compare our proposed tree-based methods (Ours-Best and Ours-Vote) against several baselines:

- Vote (Wang et al., 2023b): Standard majority voting among N outputs.
- Length-Best (Wang et al., 2025): Select the response with the fewest tokens.
- Length-Vote (Wu et al., 2025): Majority voting after selecting the k responses with reliable CoT length.
- ORM-Best (Brown et al., 2024): Select the response with the highest outcome reward model score using Skywork-Reward-Gemma-2-27B-v0.2 (Liu et al., 2024b).
- ORM-Vote (Brown et al., 2024): Weighted Majority voting (Lightman et al., 2023) with the outcome reward model score.
- PRM-Best (Zhang et al., 2025), which scores responses based on the product of step-level scores from a process reward model (i.e., Qwen2.5-Math-PRM-72B)
- PRM-Vote (Zhang et al., 2025), Weighted Majority voting (Lightman et al., 2023) with the processing reward model score.
- Ours-Best: Select the response with the highest score assigned by our tree-based reasoning quality classifier mentioned in Section 3.3.
- Ours-Vote: Weighted Majority voting with the score of our classifier.

Our method consistently outperforms traditional heuristics and reward model-based baselines, particularly in the MATH dataset, where precise multi-step reasoning is crucial. Notably, for DeepSeek-32B on MATH, our tree-based voting method achieves the highest accuracy at 82.47%, significantly surpassing both Length-Best (56.70%) and ORM-Best (60.82%). Similar trends are observed for QwQ-32B, with our model showing competitive or superior performance. These results confirm that incorporating structural reasoning patterns via LCoT2Tree leads to a reliable output selection in complex reasoning tasks.

C.4 Comparison with Additional Baselines

To further contextualize our approach, we evaluate against two additional baselines: (1) **Tree Construction with Simple Prompt**, following the tree structure in Tree of Thoughts by prompting DeepSeek-V3 to directly generate JSON-formatted trees, which are then classified using the same GNN as our method; and (2) **LLM-based Classifier**, where we fine-tune a classification head on top of a frozen Qwen-2.5-Math-7B and Llama-3.1-8B-Instruct backbone. The experimental setting is the same with Section 3.3.

As shown in Table 7, our approach achieves comparable or better performance than LLM-based classifiers, despite not leveraging full semantic content. Compared to one-step tree generation, our method achieves much higher accuracy, as prompted tree outputs often lack stability and detail when dealing with complex reasoning. These results further demonstrate the effectiveness of our structured approach.

C.5 Cost-Aware Evaluation and Prompted Trees

To evaluate the fairness of comparing our method with simple heuristics (e.g., response length), we consider both accuracy and computational cost

Method	MATH	LCB	GPQA	MMLU
Tree via Prompt	0.6628	0.6121	0.5678	0.5556
Qwen-2.5 + Classification Head	0.7207	0.8035	0.6658	0.7089
Llama-3.1 + Classification Head	0.7638	0.8109	0.6658	0.7090
Ours	**0.8081**	**0.8221**	**0.7037**	**0.7241**

Table 7: Comparison with additional tree-based and LLM-based baselines. Our structural representation achieves consistently strong performance.

(measured in input/output tokens). We further implement an additional baseline, **Tree via Prompt**, which directly prompts the model to output a structured reasoning tree in JSON format. While this approach avoids the multi-stage pipeline, it proves unstable and often omits critical details when reasoning content is long, leading to reduced accuracy.

Despite the higher cost, our method substantially outperforms both length-based and prompt-based baselines (e.g., +12.5% accuracy on MMLU). This highlights the value of structural reasoning. To mitigate token cost, we plan to explore fine-tuning models with our generated reasoning trees as supervision, enabling one-step structured reasoning generation in the future.

D Diagnostic Insight into Reasoning Behaviors & Visualization Results

D.1 Insight into Error Behaviors

In this section, we present a detailed analysis of common error patterns found within reasoning trees. We use GNNExplainer (Ying et al., 2019), a graph-based interpretability method, to identify which edges in a reasoning tree contribute most significantly to the model's predictions. For each reasoning tree, GNNExplainer assigns an importance weight to every edge, reflecting its influence on the model's output. These weights are normalized to the $[0, 1]$ range, and we visualize the tree by adjusting the edge thickness and color intensity according to these scores. The darker and thicker the edge, the more critical it is to the model's decision. Illustrative examples are shown in Figure 9.

Based on this analysis, we extract and categorize the most usual subgraphs associated with incorrect predictions into four primary error patterns. (A) Over Branching: excessive exploration or verification from a single node; (B) Step Redundancy: repetitive or unnecessary reasoning within the same step; (C) Direct Reasoning: abrupt transitions from one reasoning step to much deeper steps with minimal branching; (D) Skipped Thinking: leaping across multiple reasoning steps without proper intermediate logic.

These patterns are visualized in the left part of Figure 9, with real examples provided on the right. Notably, these findings reveal that both overly complex and overly simplistic reasoning paths can lead to incorrect outcome, underscoring the need for balanced, coherent, and well-structured reasoning in high-quality LLMs.

D.2 Task-specific Reasoning Behaviors

We have quantitatively demonstrated that LCoT2Tree effectively facilitates the separation of task-specific reasoning contents, as detailed in Section 4.2. In this section, we leverage LCoT2Tree to pinpiont the disparate behaviors exhibited by the DeepSeek-32B model across various tasks. The key findings are summarized below:

For MATH (Figure 10), the reasoning trees typically display a diagonally descending structure, with progressively deeper steps achieved through repeated backtracking. This pattern reflects a structured, hierarchical problem-solving strategy. In the visualization, dashed lines—representing backtracking—are identified as key structural features that distinguish MATH from other tasks.

For LiveCodeBench (Figure 11), the trees often exhibit broad, parallel branching, where many sibling nodes continue with independent linear thoughts that are rarely explored or verified further. This suggests a shallow, scattered reasoning style. Our visualization also reveals that these parallel branches contribute most significantly to classifying this task.

For GPQA (Figure 12), the reasoning trees contain numerous high out-degree nodes, indicating that the model frequently revisits and expands on specific concepts. This behavior suggests intensive cognitive effort and repeated clarification, reflecting the model's attempt to thoroughly understand difficult points—while also hinting at a lack of confidence in its reasoning.

Dataset	Method	Accuracy	Input Tokens	Output Tokens
MMLU	Length	0.5995	0	0
	Tree via Prompt	0.5556	2.7K	0.8K
	Ours	**0.7241**	14.5K	1.3K
MATH	Length	0.7413	0	0
	Tree via Prompt	0.6628	6.0K	0.8K
	Ours	**0.8081**	33.2K	1.7K

Table 8: Accuracy and token cost comparison across methods. Our approach achieves significantly higher accuracy despite higher token usage.

Finally, for MMLU-Pro(Figure 13), the reasoning trees are relatively shallow, with fewer nodes and minimal branching. This suggests a more direct, deductive approach with limited exploration, which is consistent with the knowledge-intensive nature of MMLU-Pro questions rather than deeply compositional reasoning.

These observations highlight how LCoT2Tree provides fine-grained insights into the cognitive strategies employed by the model in diverse reasoning scenarios.

D.3 Model-specific Reasoning Behaviors

We provide a detailed comparison of how different LLMs approach the same task by visualizing and analyzing their reasoning trees on the MATH dataset. Focusing on DeepSeek-32B as a reference point, we summarize several key observations:

DeepSeek-32B (DS-32; Figure 10) typically produces reasoning trees with a diagonally descending structure, with depth increasing progressively through backtracking. This reflects a structured, step-by-step problem-solving reasoning process.

DeepSeek-R1 (Figure 14) exhibits similar structural characteristics to DS-32, but with a notable difference: it tends to terminate detailed exploration earlier and backtrack more quickly to beginning steps. This indicates a more aggressive pruning strategy to streamline the reasoning path. In visualizations, connections between Step 0 and Step 1 serve as critical features distinguishing DeepSeek-R1's behavior.

QwQ-32B (Figure 15) also mirrors the behavior of DS-32 to some extent but differs in the latter stages. Unlike DS-32, which often rushes toward the final answer, QwQ-32B continues to invest cognitive effort into deeper exploration. In the visualization, expanded right subtrees often emerge as defining characteristics of QwQ-32B's reasoning tree.

In contrast, Seed-1.5-Thinking-pro (Figure 16) and Grok-3-mini-beta (Figure 17) follow a markedly different reasoning strategy. They exhibit fewer thought transitions during reasoning. As a result, their trees contain fewer nodes and branches, forming simpler structures. This suggests a straightforward problem-solving style with limited iterative refinement.

These insights reinforce that LCoT2Tree not only captures reasoning structure at the task level, but also reveals distinctive behavioral patterns across model families.

Step1 Prompt in LCoT2Tree tool to extract reasoning sketch from LCoT

Analyze the following reasoning text and extract a strictly ordered, atomic sequence of key reasoning steps. Focus on extracting the validated, logically essential progression of thoughts while excluding backtracking, rechecks, or redundant details.

Reasoning text:
<reasoning_text>
{{text}}
</reasoning_text>

Please read the entire text carefully and generate by following these rules:
1. Find the key steps and the logical flow of reasoning.
2. Each step must represent a single, indivisible logical action that directly advances the reasoning.
3. Determine the correct version of the step, ignoring redundant information. A correct step should be able to push the reasoning logic forward and have no errors in itself.
4. Do not skip steps. Do not merge steps. Use the original phrasing where possible.
5. Do not include verification steps unless it introduces new constraints.
6. Organize the steps into a coherent sequence of key reasoning steps and number it sequentially (1., 2., 3., ...).
7. Maintain strict output format.

Output format:
<reasoning_process>
Step 1. concise statement: Detail step
Step 2. concise statement: Detail step
Step 3. concise statement: Detail step
</reasoning_process>

Please list the key reasoning steps of the provided text.

Figure 6: The content of Step1 Prompt in LCoT2Tree tool to extract reasoning sketch from LCoT.

Step3 Prompt in LCoT2Tree tool to assign reasoning step to each thought.

Your task is to match each reasoning thought from List B to corresponding step number(s) in the List A. Follow the following process:

1. First understand List B:
- For each thought in List B, identify if it describes some specific calculation processes (mathematical operation, logical transformation, or data manipulation)
- Ignore the describation that only state conclusions, concepts without showing the actual processing detail

2. Then math to List A:
- For each thought from List B, find all steps in List A that:
* Show the same underlying calculation (even with different numbers/words)
* Represent the partial or same reasoning process
- Ignore superficial wording differences - focus on logical equivalence

3. Output requirements:
- Return ALL plausible matches where computational processes align
- Never return empty arrays (except for thought B0 if needed)
- Multiple matches are encouraged when justified
- Maintain strict JSON format

Input:
- List A (Detailed Steps):
<list_a>
{{reasoning_step}}
</list_a>
- List B (Reasoning Thoughts):
<list_b>
{{thoughts}}
</list_b>

Output Format (strict JSON):
```json
{
"B0": ["A1"],
"B1": ["A3"],
"B2": ["A1", "A4"],
...
}
```

Please match the reasoning thoughts in List B to step in the List A.

Figure 7: The content of Step3 Prompt in LCoT2Tree tool to assign reasoning step to each thought.

Step4 Prompt in LCoT2Tree tool to assign function to each thought.

Your task is to classify Text2's purpose relative to Text1 using these categories:

Categories:
1. Continuous Logic - Direct continuation/extension of Text1's reasoning flow
2. Exploration - Introduces parallel/unrelated concepts from Text1, alternative reasoning paths, or new topics
3. Backtracking - Revises, corrects, or adjusts previous step
4. Validation - Provides supporting evidence, logical justification, or examples for Text1's claims

Input: {{
"Text1": "TEXT1",
"Text2": "TEXT2"
}}

Output Format:
Return only JSON format "'json{"Category": "Name of Category"}"'

Figure 8: The content of Step4 Prompt in LCoT2Tree tool to assign function to each thought.

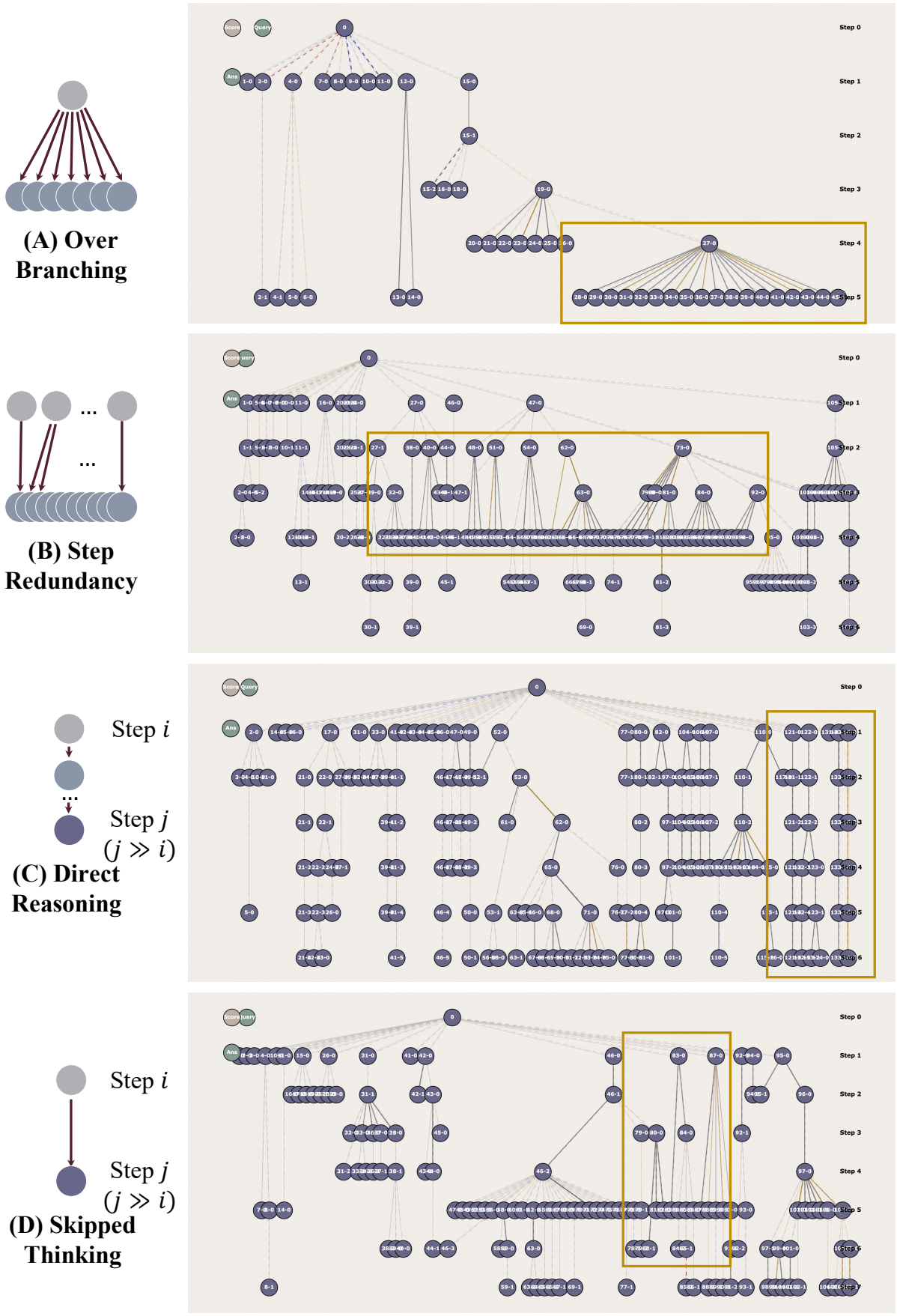

Figure 9: Visualization results of tree structure corresponding to different error patterns. The edge is labeled with the importance generated by GNNExplainer. The darker the color and the thicker the edge, the more important it is.

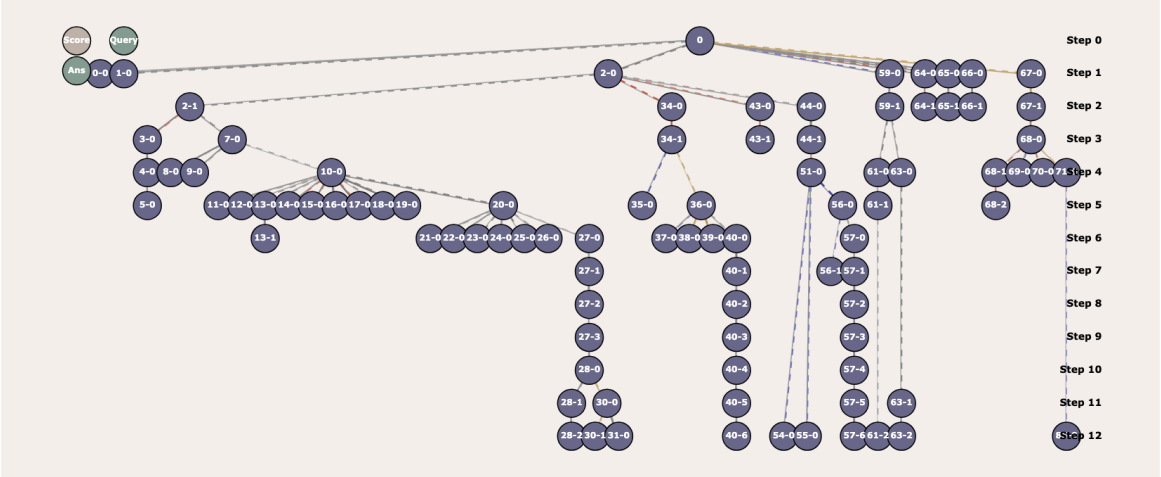

Figure 10: Visualization results of tree structure of a response from **DeepSeek-32B** on **MATH** dataset extracted using LCoT2Tree. The reasoning trees exhibit a downward-sloping hierarchical structure, with progressively deeper steps achieved through repeated backtracking.

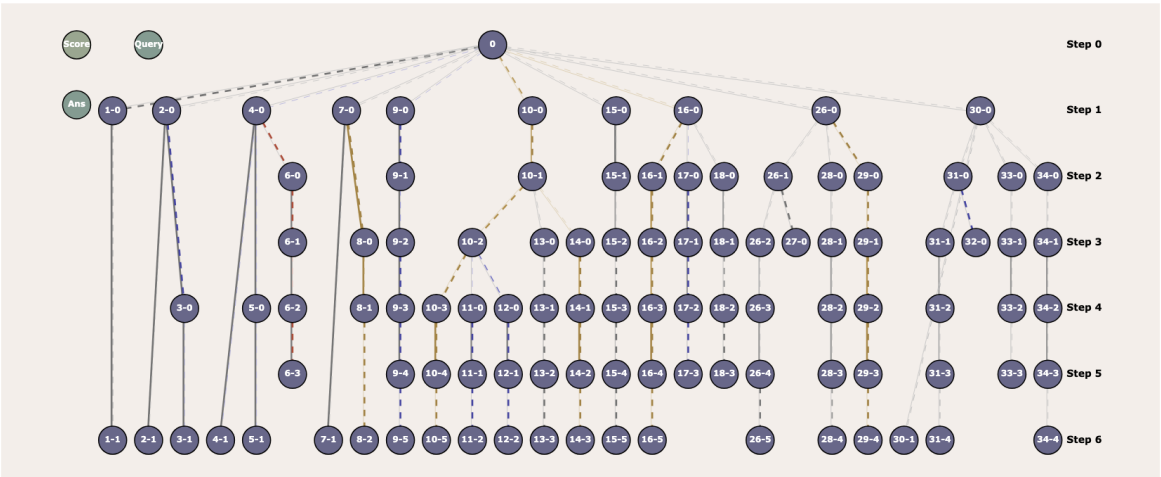

Figure 11: Visualization results of tree structure of a response from **DeepSeek-32B** on **LiveCodeBench** dataset extracted using LCoT2Tree. The reasoning patterns tend to show broad, parallel branching, where many sibling nodes initiate independent linear thought without subsequent exploration or verification.

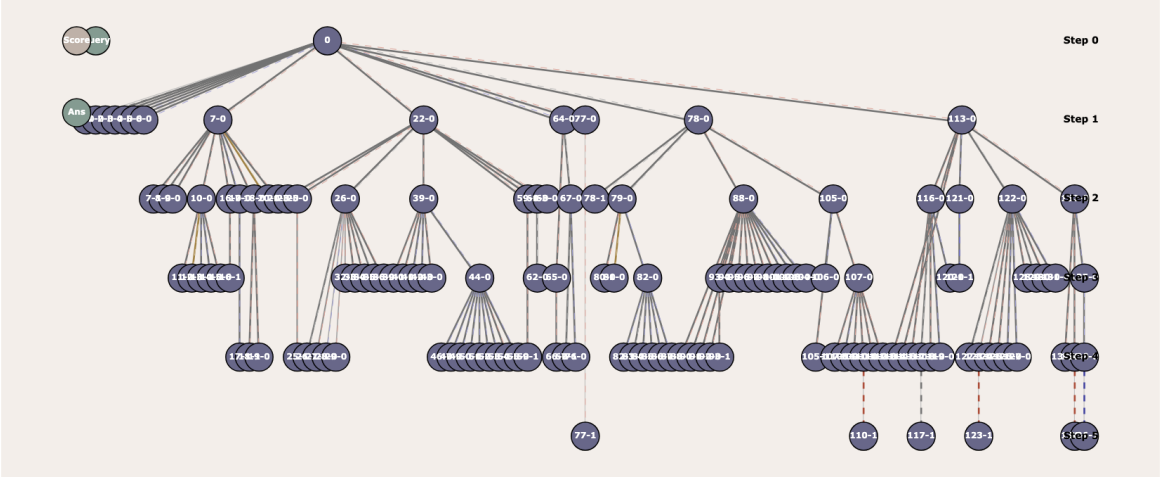

Figure 12: Visualization results of tree structure of a response from **DeepSeek-32B** on **GPQA** dataset extracted using LCoT2Tree. The reasoning trees contain many high out-degree nodes, indicating that the model often revisits and elaborates on complex concepts.

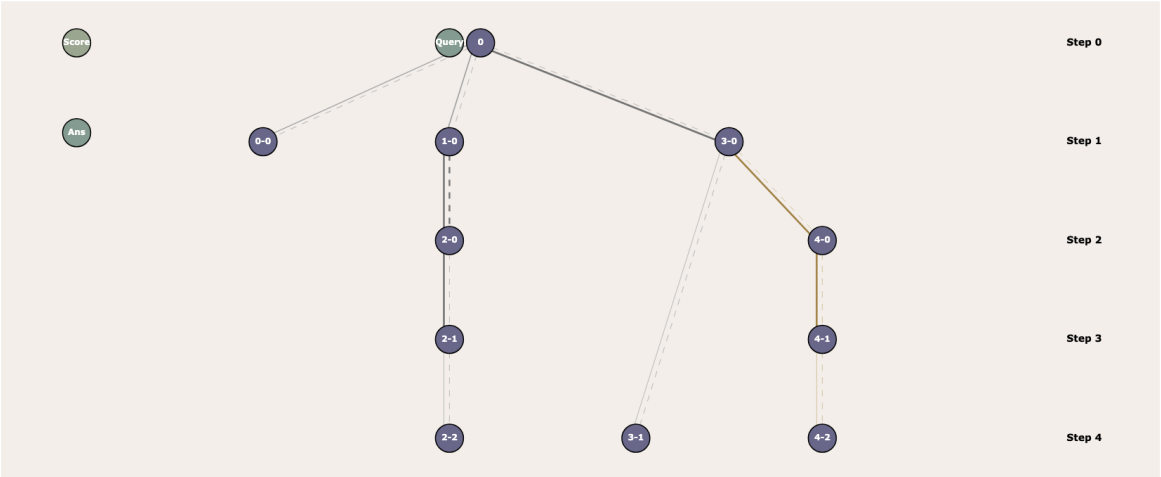

Figure 13: Visualization results of tree structure of a response from **DeepSeek-32B** on **MMLU-Pro** dataset extracted using LCoT2Tree. The reasoning trees contain fewer nodes and minimal branching, indicating a more direct and deductive reasoning style with less exploration.

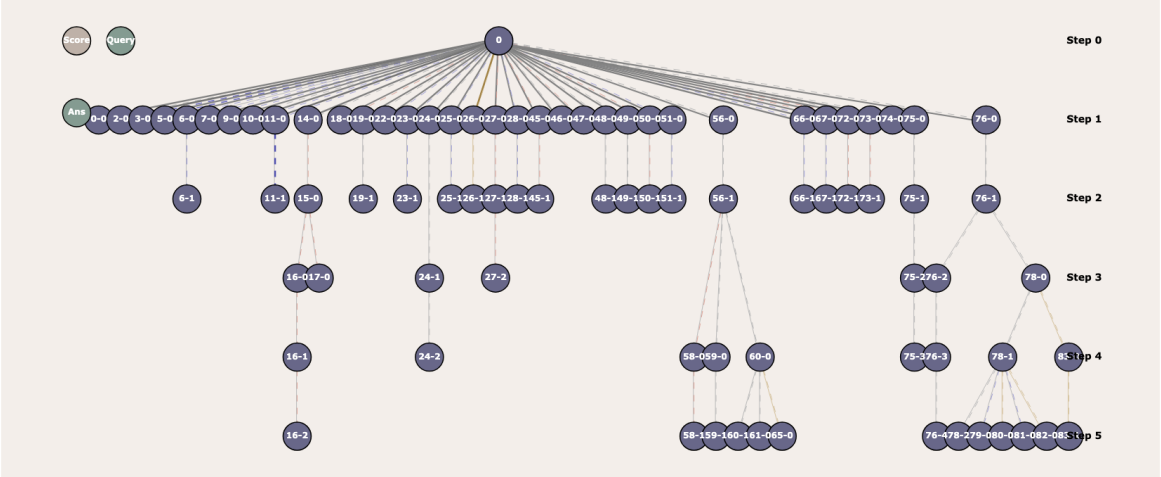

Figure 14: Visualization results of tree structure of a response from **DeepSeek-R1** on **MATH** dataset extracted using LCoT2Tree. It exhibits similar behavior to DS-32, but with an important distinction: it tends to truncate detailed exploration earlier and backtrack to beginning steps more quickly to optimize its reasoning path.

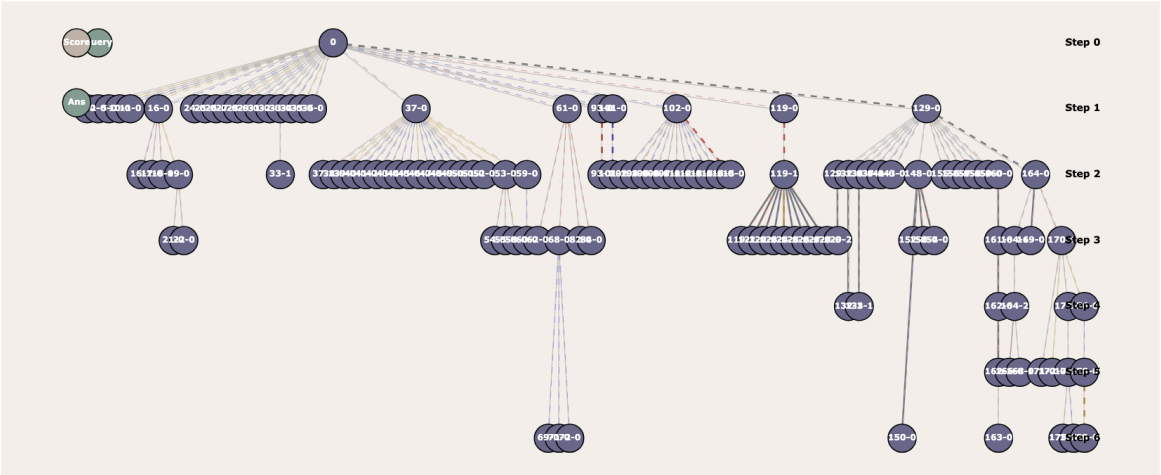

Figure 15: Visualization results of tree structure of a response from **QwQ-32B** on **MATH** dataset extracted using LCoT2Tree. QwQ-32B mirrors the behavior of DeepSeek-32B to some extent, but differs in how it allocates attention in the latter stages of reasoning.

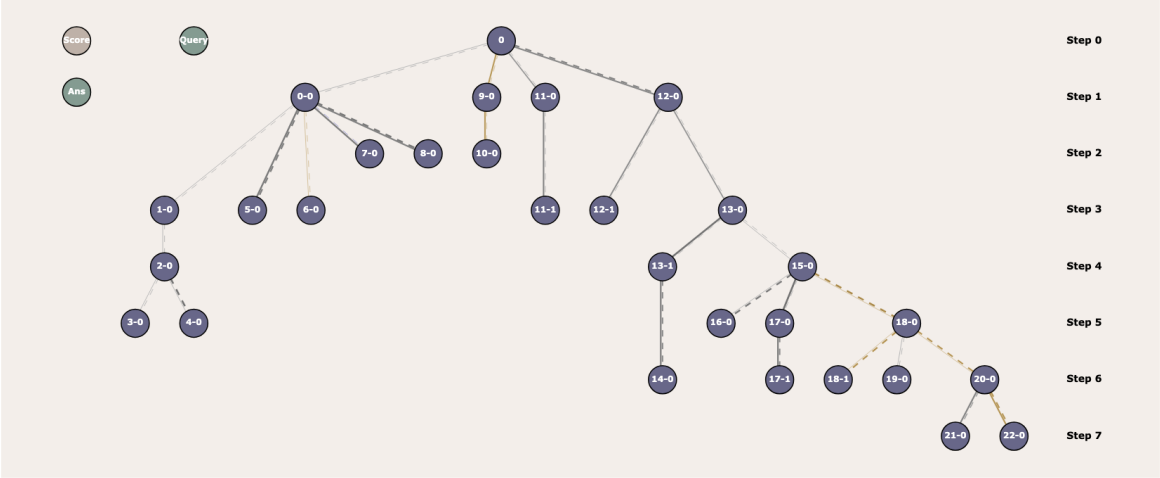

Figure 16: Visualization results of tree structure of a response from **Seed-1.5-Thinking-pro** on **MATH** dataset extracted using LCoT2Tree. The reasoning trees contain fewer nodes and branches, forming simpler structures.

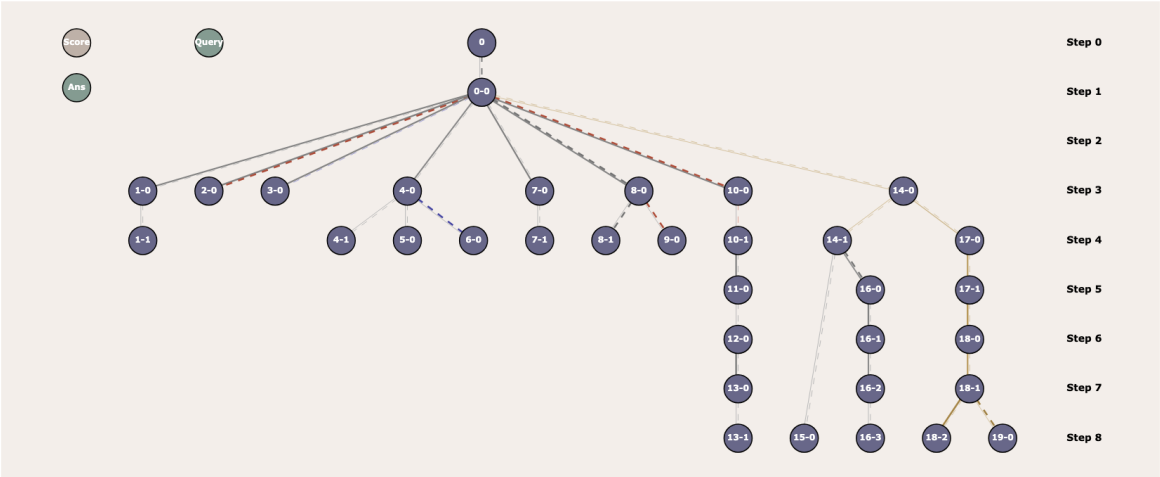

Figure 17: Visualization results of tree structure of a response from **Grok-3-mini-beta** on **MATH** dataset extracted using LCoT2Tree. The reasoning trees contain fewer nodes and branches, forming simpler structures.