RamPO: Retrieval-Augmented Monte Carlo Tree Search Preference Optimization for Multi-Hop Question Answering

Anonymous ACL submission

Abstract

Large language models achieve impressive performance on NLP tasks. Nevertheless, multi-hop question answering (QA) requires exploring a vast search space of possible reasoning, which leads to performance degradation. Recent methods often enhance multi-hop reasoning relying on inference scaling laws (e.g., performing multiple rollouts to enhance reasoning), which significantly increases latency; or retrieval-augmented generation (RAG), which requires additional valid reasoning over the retrieved content. However, real-time QA scenarios demand LLMs explore the search space for valid reasoning within limited time budgets. In this work, we propose **R**etrieval-**A**ugmented **M**onte Carlo Tree Search **P**reference **O**ptimization (**RamPO**), which integrates Monte Carlo Tree Search (MCTS) with a comprehensive action sequence tailored for RAG settings. By leveraging MCTS-guided heuristic exploration to constrain the search space and aligning with preference preferred by MCTS in offline, RamPO provides a trade-off between latency and reasoning accuacy during search space exploration in online inference. Experiments in three multi-hop QA datasets show that RamPO achieves an average performance improvement of 12.3% compared to recent top-notch methods, with up to 156.2× faster than existing tree-like inference approaches. Our code is available at `https://github.com/NLPwang/RamPO`.

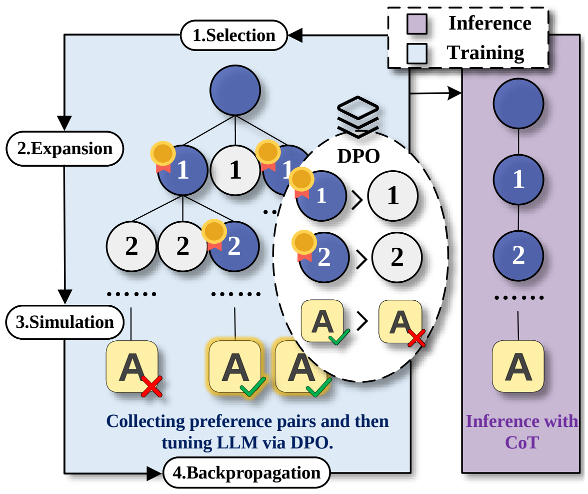


Figure 1: An overview of RamPO, which leverages MCTS with a comprehensive action sequence tailored for RAG settings to guide heuristic exploration of search space for the construction of preference pairs, and employs DPO to train LLMs.

1 Introduction

Recent advances in large language models (LLMs) (Ouyang et al., 2022; Achiam et al., 2023; Touvron et al., 2023) have demonstrated strong capabilities in solving downstream tasks (Brown et al., 2020; Raffel et al., 2020). However, multi-hop question answering (QA) (Yang et al., 2018; Ho et al., 2020; Press et al., 2022) remains particularly challenging, as it requires LLMs to integrate and reconcile evidence across reasoning steps and multiple sources before arriving at a final answer.

Several efforts have sought to enhance multi-hop reasoning relying on inference scaling laws (ISL) (Wu et al., 2024) , where increased computational resources or extended inference time are used to boost reasoning quality (Yao et al., 2023a; Hao et al., 2023; Feng et al., 2023; Zhang et al., 2024a). ISL substantially raises latency, limiting its applicability in real-time scenarios. To reduce inference latency, recent work (Zhang et al., 2024c) attempts to shift latency from inference to training via reinforcement learning (RL) (Rafailov et al., 2023; Guan et al., 2025). However, data collected for RL typically relies on tree-of-thought (ToT) (Yao et al., 2023a) combined with BFS or DFS. Within limited time budgets, it remains difficult to comprehensively explore the immense search space and explore valid reasoning.

Method	Reasoning Policy	Inference	Retrieval-Augmented
CoT (Wei et al., 2022b)	CoT	Chain-Like	🪫
RQ-RAG (Chan et al., 2024)	CoT		🔋
CPO (Zhang et al., 2024c)	ToT		🪫
ToT (Yao et al., 2023a)	ToT	Tree-Like	🪫
ReST-MCTS* (Zhang et al., 2024a)	MCTS		🪫
RamPO (Ours)	MCTS	**Chain-Like**	🔋

Table 1: Comparison between the related top-notch baseline methods and our proposed RamPO. 🔋 : fully support, 🪫 : partial support (i.e., performance may remain stable or even decline when adapting the standard RAG in the follow-up experiment).

Some methods incorporate retrieval-augmented generation (RAG) (Trivedi et al., 2023) into the reasoning process, enabling LLMs to leverage external knowledge during QA. Rather than solely relying on the initial query for retrieval, recent work enhances query understanding by applying preprocessing operations, enabling the retrieval of more relevant documents (Chan et al., 2024; Zhang et al., 2024b; Hu et al., 2025). However, external knowledge introduces additional challenges: models must not only identify relevant information from multiple documents but also reason over it to derive correct answers. Unguided and unstructured exploration becomes highly challenging when the search space of possible reasoning is immense.

To provide a heuristic and structured search strategy for exploration, we propose **R**etrieval-**A**ugmented **M**onte Carlo Tree Search **P**reference **O**ptimization (**RamPO**), which integrates Monte Carlo Tree Search (MCTS) with a comprehensive action sequence tailored for RAG settings. As shown in Figure 1, RamPO leverages MCTS and answer correctness as reward signals to guide heuristic exploration of search space for the construction of preference pairs. A preference pair respectively consists of the preferred step and the dispreferred step, both of which share the same previous reasoning. Subsequently, RamPO employs direct preference optimization (DPO) (Rafailov et al., 2023) to train LLMs, aiming to align with preference and shift the latency from inference to training. As shown in Table 1 and Figure 2, compared to previous methods, RamPO enables LLMs to follow the MCTS-preferred reasoning through a single rollout with CoT in inference, which provides a trade-off between latency and reasoning accuacy during the search space exploration.

Our key insight is that much like strategic planning in games (e.g., chess) or navigating a maze, MCTS simulates complete future reasoning based on the current steps before committing to the next step, which provides heuristic and structured exploration of the search space, enabling informed decision-making at each step rather than relying on random or unstructured sampling.

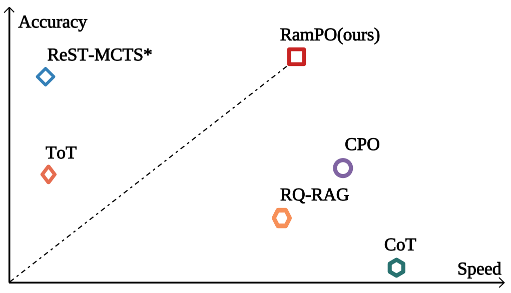


Figure 2: RamPO provides a trade-off between latency and reasoning accuacy during search space exploration in online inference.

RamPO exhibits exceptional performance across three mainstream LLMs on three commonly used complex multi-hop QA datasets. Compared to recent top-notch methods, after preference optimization, it achieves an average accuracy improvement of up to 12.3% and a maximum improvement of 36.2%. While demonstrating comparable or superior performance to tree-like inference methods, RamPO also shows lower latency, being 156.2× faster than tree-like inference methods.

Our contributions are as follows:

- We integrate MCTS with a comprehensive action sequence for RAG settings into reasoning for multi-hop QA, providing heuristic and structured exploration of the search space. RamPO provides a trade-off between latency and reasoning accuacy during search space exploration in online inference within limited time budgets.
- We provide empirical evidence that RamPO enhances the reliability and robustness of reasoning in LLMs by heuristically exploring and evaluating the search space.

2 Related Work

Reasoning Augmentation for LLMs. Recent advances in LLMs emphasize constructing reasoning chains to enhance problem-solving capabilities. Chain-of-Thought (CoT) (Wei et al., 2022b) improves reasoning by generating intermediate steps. Building on this, Yao et al. (2023a) propose Tree-of-Thought (ToT), which enables a more deliberate reasoning process by exploring multiple branching thoughts at each step. Subsequent work further enhances the tree-like reasoning through advanced tree-search algorithms such as Monte Carlo Tree Search (MCTS) incorporating external reward models (Feng et al., 2023; Zhang et al., 2024a), which still requires tree-like reasoning during the inference phase. To reduce ToT's high latency, Tian et al. (2024) distill ToT's strategic depth into CoT by fine-tuning LLMs using optimal reasoning identified by ToT. Extending this, Zhang et al. (2024c) apply Direct Preference Optimization (DPO) (Rafailov et al., 2023) with paired preference thoughts gathered in ToT, leveraging additional preference information in dispreferred thoughts.

However, within limited time budgets, it remains difficult to comprehensively explore the immense search space relying on ToT with BFS or DFS. Meanwhile, chain-like and tree-like reasoning is inherently static and blackboxed (Yao et al., 2023b), as it depends solely on internal representations without grounding in external knowledge, limiting its adaptability in complex multi-hop question answering.

Retrieval-Augmented Generation for LLMs. RAG (Lewis et al., 2020) enhances LLMs in knowledge-intensive tasks by integrating external information into the generation process. Trivedi et al. (2023) facilitate reasoning over retrieved documents by integrating CoT reasoning. Chan et al. (2024) focus on optimizing the retrieval phase through query rewriting and decomposition. Furthermore, recent research has explored combining tree-like reasoning structures with RAG to further enhance the flexibility of search space exploration (Zhang et al., 2024b). Hu et al. (2025) integrates MCTS's reasoning and search capabilities with adaptive retrieval mechanisms.

However, recent work typically employs inference scaling laws (e.g., tree-like reasoning) to achieve more deliberate reasoning in RAG scenarios, which significantly increases latency and struggles to meet real-time requirements.

In this work, our proposed **RamPO** uses answer correctness as reward signals to explore the search space guided by MCTS and construct preference pairs. Subsequently, LLMs are trained to align with preference via DPO, shifting the latency from inference to training. RamPO enables LLMs to follow the MCTS-preferred reasoning through a single rollout with CoT in inference by leveraging MCTS to guide heuristic exploration. Meanwhile, RamPO introduces a comprehensive action sequence framework during MCTS for RAG scenarios. This facilitates the integration of the RAG mechanism with the MCTS reasoning structure, thereby improving adaptation to challenging multi-hop QA scenarios.

3 Preliminary

To facilitate a better understanding of our proposed RamPO, we introduce the mechanism of **Monte Carlo Tree Search (MCTS)** within this section.

MCTS (Browne et al., 2012) is a tree-based decision-making algorithm widely used in complex decision processes, which balances exploration and exploitation. MCTS maintains a search tree that explicitly records historical trajectories and associated statistical metrics, comprising four essential phases:

- **Selection**: Traverse the existing search tree from the root node, recursively selecting optimal child nodes based on specific criteria (e.g., **Upper Confidence Bound applied to Trees, UCT**). This phase continues until it encounters either an expandable leaf node (a node that has not been fully expanded) or a terminal node (e.g., terminating at the maximum depth or arriving at the final answer). The UCT is calculated as follows:

$$\text{UCT}(s_i, s) = \frac{Q_{s_i}}{N_{s_i}} + c\sqrt{\frac{\log(N_s)}{N_{s_i}}} \quad (1)$$

  where Q_{s_i} and N_{s_i} represent the estimated value and the visit count of node s_i respectively. While N_s denotes the visit count of the parent node s of s_i. c is a constant that balances the trade-off between exploration and exploitation. Specifically, c is set to $\sqrt{2}$ in this work.

- **Expansion**: Expand the current node when reaching an expandable leaf node rather than a terminal node, by adding a feasible child node selected based on the action of the current node.

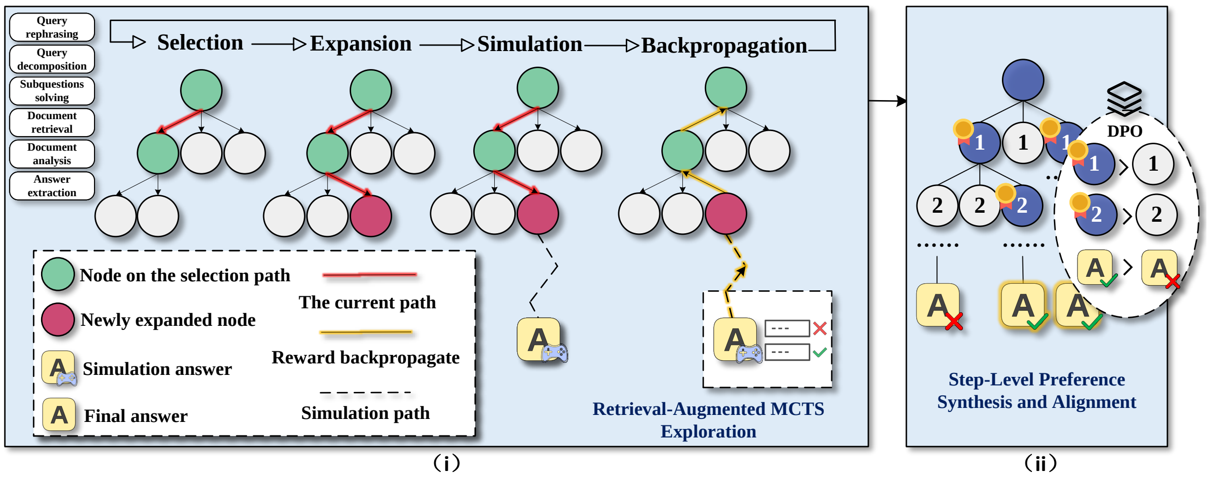


Figure 3: The framework of RamPO. (i) illustrates the exploration of the search space, guided by MCTS, utilizing a comprehensive action sequence specifically designed for RAG settings. (ii) depicts the process of collecting preference pairs and employing them to optimize the model. Nodes marked with a medal icon indicate preferred reasoning path that lead to correct answers. After optimization, the model requires only a single rollout during inference to obtain the preferred reasoning path, resulting in low-latency execution.

- **Simulation**: Perform simulations (namely rollouts) with a specific policy (e.g., random policy in this work) to estimate the estimated value Q_{s_i} of the newly expanded node, continuing until a terminal node is reached.
- **Backpropagation**: Backpropagate the **Simulation** result to the root along the path. The statistical metrics (visit counts N, estimated values Q) of all visited nodes along the path are updated based on the reward of the simulation terminal node (calculated by a custom reward function).

By iteratively executing these four phases, MCTS effectively balances exploration and exploitation, thereby optimizing strategies in scenarios where the vast search space renders exhaustive exploration impractical.

4 RamPO

As shown in Figure 3, RamPO consists of two main components: Retrieval-Augmented MCTS Heuristic Exploration for heuristic exploration of search space in RAG scenarios under MCTS, and Step-Level Preference Synthesis and Alignment for gathering preference pairs and training LLMs to align with preference using DPO.

4.1 Retrieval-Augmented MCTS Heuristic Exploration

We establish a seamless integration between MCTS in §3 and RAG. The step-wise exploration and generation process adheres to the standard phases of MCTS, where each node in the search tree corresponds to a solution step generated by the LLM, conditioned on all previously generated steps along the current reasoning. After the search tree is fully constructed via iterative exploration, we extract preference pairs from the resulting paths.The details are as follows:

Reasoning Action Sequence Under RAG. To enhance the adaptability of the MCTS algorithm with RAG, we design a comprehensive action sequence framework that systematically integrates RAG concepts into the search tree construction process, which comprises six distinct operational actions:

◇ A_1: Query rephrasing (Ma et al., 2023).
◇ A_2: Query decomposition (Zhou et al., 2023).
◇ A_3: Subquestions solving (Zhou et al., 2023).
◇ A_4: Document retrieval (Ram et al., 2023).
◇ A_5: Document analysis (Wei et al., 2022a).
◇ A_6: Answer extraction (Wei et al., 2022a).

We define the action sequence:

$$\mathbf{S} = (A_1, A_2, A_3, A_4, A_5, A_6) \qquad (2)$$

where A_1, A_2 are designed for query optimization, aiming to enhance the comprehension of the query and improve the accuracy of relevant document retrieval. A_3, A_5, A_6 are actions for analysis and reasoning built upon previous actions and retrieved documents. A_4 focuses on document retrieval.

At each step i, MCTS selects and executes an action a_i following $\mathbf{S}$. a_i is used to prompt the LLM to sample the child node s_i based on the current reasoning path $r \oplus s_1 \oplus s_2 \oplus \ldots \oplus s_{i-1}$, where r represents the root, i.e., the user query, and $s_j (1 \leq j < i)$ denotes a reasoning step (node) generated by the LLM. More details on action-related prompts can be found in Appendix B.

Reasoning Space Exploration with MCTS. We iteratively explore the search space by carrying out the MCTS process detailed in §3. Starting from the root node (the user query), MCTS recursively selects promising child nodes based on the UCT. This selection phase continues until an expandable leaf node s_{i-1} is encountered. Once an action is chosen based on $\mathbf{S}$, a child node s_i sampled by the LLM is subsequently expanded. The estimated value Q of the newly expanded node s_i is then estimated through a simulation, as described in the following paragraph.

Reward Evaluation and Backpropagation. To calculate the estimated value Q of each node, we adopt a method inspired by AlphaGo (Silver et al., 2017), where each intermediate node is evaluated based on its contribution to the final correct answer. Specifically, actions that more frequently lead to the correct answer are assigned higher scores, increasing their likelihood of being selected during the tree expansion process.

Following the expansion, a simulation is conducted to estimate its initial estimated value and the reward. The simulation leverages the LLM to generate corresponding steps in accordance with $\mathbf{S}$, continuing until the final step A_6 (Answer extraction) is reached. Subsequently, we employ **Hard Estimation** to evaluate the reward r and assign the initial estimated value $Q_{s_i}^{initial}$ to the newly expanded node s_i, as defined below:

$$Q_{s_i}^{\text{initial}} = r_{s_i} = \begin{cases} 1, & A_{simulation} = GT \\ 0, & \text{else} \end{cases} \tag{3}$$

where $A_{simulation}$ denotes the answer generated by the LLM with the steps from the root to the newly expanded node s_i in this round of simulation, and GT represents the ground truth answer for the given problem.

The reward r_{s_i} is then backpropagated along the reasoning path $P = r \oplus s_1 \oplus s_2 \oplus ... \oplus s_{i-1}$, such that the estimated value of each intermediate node s_j is updated as $Q_{s_j} \leftarrow Q_{s_j} + r_{s_i}$.

4.2 Step-Level Preference Synthesis and Alignment

Preferences Synthesis. As shown in Figure 3(ii), once the tree has been fully constructed through iterative exploration, the nodes along the paths that lead to the correct answer are identified as preferred (i.e., winning) nodes. For each preferred node s_i^w, corresponding dispreferred (i.e., losing) nodes s_i^l are derived. This is accomplished by first locating the parent node s_{i-1}^w. Among the child nodes of s_{i-1}^w, those that are not marked as preferred are considered candidates for the dispreferred nodes s_i^l, which naturally provides additional preference information. Crucially, only preferred and dispreferred nodes that share the same parent node s_{i-1}^w are eligible to form a valid preference pair (s_i^w, s_i^l). To prevent external documents from introducing noise during training, we deliberately remove the original retrieved documents when constructing preference pairs.

Preferences Alignment. Once we have obtained the Step-Level preference pairs, we leverage recent advances in reinforcement learning from human feedback (RLHF), particularly the Direct Preference Optimization (DPO) algorithm (Rafailov et al., 2023), to align LLMs with the preference collected in MCTS. Specificly, for the i-th step, given the previous reasoning steps $s_{1:i-1}^w$, the probabilities of generating the preferred reasoning step s_i^w and the dispreferred reasoning step s_i^l are denoted $\pi_\theta(s_i^w|x, s_{1:i-1}^w)$ and $\pi_\theta(s_i^l|x, s_{1:i-1}^w)$, respectively. To optimize the LLM on this pair of preference steps, we can directly substitute it into the DPO framework:

$$\begin{aligned}\mathcal{L}_i(\pi_\theta; \pi_{\text{ref}}) = -\log \sigma \Bigg(& \beta \log \frac{\pi_\theta(s_i^w|x, s_{i-1}^w)}{\pi_{\text{ref}}(s_i^w|x, s_{i-1}^w)} \\ & -\beta \log \frac{\pi_\theta(s_i^l|x, s_{i-1}^w)}{\pi_{\text{ref}}(s_i^l|x, s_{i-1}^w)} \Bigg)\end{aligned} \tag{4}$$

where σ is the logistic function, the hyperparameter β regulates the penalty imposed for the deviations from the base reference model π_{ref}.

5 Experiments

5.1 Datasets

We focus our research on three commonly used multi-hop QA datasets: 1) Bamboogle (Press et al., 2022), comprising complex questions that Google answers incorrectly, assessing models' compositional reasoning across different domains. 2) Hot-

	CoT		RQ-RAG		CPO		ToT		ReST-MCTS*		**RamPO (ours)**	
	Acc.$\uparrow$ (%)	Lat.$\downarrow$ (s/ins.)	Acc.$\uparrow$ (%)	Lat.$\downarrow$ (s/ins.)	Acc.$\uparrow$ (%)	Lat.$\downarrow$ (s/ins.)	Acc.$\uparrow$ (%)	Lat.$\downarrow$ (s/ins.)	Acc.$\uparrow$ (%)	Lat.$\downarrow$ (s/ins.)	Acc.$\uparrow$ (%)	Lat.$\downarrow$ (s/ins.)
LLaMA-3.2-3B-Ins												
Bam.	30.4(2.9$\uparrow$)	2.0	41.1	3.0	33.9(0.1$\downarrow$)	3.1	34.3(0.2$\downarrow$)	764.6	41.5(1.9$\downarrow$)	665.9	45.3*	5.0
Hot.	21.6(6.6$\uparrow$)	2.3	24.9	2.9	28.5(4.2$\uparrow$)	3.3	26.4(1.2$\uparrow$)	802.3	34.1(3.2$\downarrow$)	705.2	38.3*	5.2
2Wiki.	22.0(1.0$\downarrow$)	2.3	13.5	3.3	24.5(3.4$\uparrow$)	2.9	24.8(3.3$\uparrow$)	832.1	32.7(0.2$\downarrow$)*	735.7	30.0	5.9
LLaMA-3.1-8B-Ins												
Bam.	55.6(3.3$\downarrow$)	4.9	58.7	5.3	62.3(13.6$\downarrow$)	3.9	64.3(9.2$\downarrow$)	1534.9	82.8(3.5$\downarrow$)*	1035.6	81.2	7.2
Hot.	28.4(5.8$\uparrow$)	4.5	34.9	5.8	34.0(4.7$\uparrow$)	4.7	31.4(2.1$\downarrow$)	1762.6	40.6(1.8$\downarrow$)	1106.7	42.9*	7.6
2Wiki.	26.5(3.5$\uparrow$)	5.8	26.7	6.4	29.2(2.4$\uparrow$)	4.1	29.7(1.0$\downarrow$)	1203.4	37.9(0.1$\downarrow$)*	1099.5	36.5	8.3
Qwen-2.5-7B-Ins												
Bam.	32.3(3.1$\downarrow$)	5.3	44.2	5.9	30.7(2.1$\downarrow$)	5.3	31.2(1.3$\uparrow$)	1263.2	69.2(1.1$\downarrow$)*	1146.1	68.5	8.2
Hot.	18.8(7.7$\uparrow$)	5.2	32.5	5.1	16.2(7.5$\uparrow$)	4.9	24.6(7.5$\downarrow$)	1596.9	38.9(0.8$\downarrow$)	1294.8	41.9*	8.0
2Wiki.	19.5(1.2$\uparrow$)	5.0	25.5	4.8	32.8(1.1$\uparrow$)	4.3	32.2(0.8$\downarrow$)	1128.0	36.4(3.8$\downarrow$)	1094.3	38.5*	7.9

Table 2: Overall performance comparison, the best results are marked with * and the best results among all **chain-like** inference methods are underlined. For methods not designed for the RAG scenario, we additionally evaluate their performance when combined with the standard RAG by appending the retrieved documents to the question. Compared to without RAG, accuracy changes are denoted by $\uparrow$ or $\downarrow$ to indicate increase or decrease respectively.

potQA (Yang et al., 2018) and 3) 2WikiMultiHopQA (Ho et al., 2020), both demanding reasoning across multiple Wikipedia paragraphs.

5.2 Baselines

We evaluate our approach against a wide range of baseline methods for comparison, categorized into chain-like and tree-like inference baselines:

- **Chain-Like Inference Baselines**:
  1) CoT (Wei et al., 2022b), which prompts the LLM to generate a series of reasoning steps before producing the final answer. 2) RQ-RAG (Chan et al., 2024), which involves query rewriting and subquestion decomposition. 3) CPO (Zhang et al., 2024c), which generates the path preferred by ToT using CoT by DPO.

- **Tree-Like Inference Baselines**:
  1) ToT (Yao et al., 2023a), which requires the LLM to explore multiple reasoning paths via tree search. 2) ReST-MCTS* (Zhang et al., 2024a), which training LLMs using model-based RL training under MCTS.

Notably, for methods not originally designed for RAG settings, we additionally evaluate their performance when combined with the standard RAG approach by appending the retrieved documents to the question. Compared to without RAG, accuracy changes are denoted by $\uparrow$ or $\downarrow$ to indicate increase or decrease respectively.

We select three widely used LLMs, specifically LLaMA-3.2-3B-Instruct, LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024) and Qwen-2.5-7B-Instruct (Yang et al., 2024). Collect the supporting statements for Bamboogle and the supporting documents for HotpotQA and 2WIKI, encode them into dense vector representations using a mainstream dense retriever (BGE-M3 (Chen et al., 2024)), and then use FAISS (Douze et al., 2024) to maintain a local vector database for retrieval. In all methods, the number of retrieved documents per question is fixed at 3 by default. Additional implementation details are presented in Appendix A.

5.3 Main Results

Table 2 summarizes the performance across various multi-hop QA datasets, from where we can conclude:

RamPO significantly enhances the capability of the LLM in solving complex multi-hop questions within the RAG scenario. As shown in Table 2, RamPO consistently outperforms all chain-like inference baselines across all datasets and models, achieving an average improvement of 14.5% and a maximum improvement of 36.2%. This indicates that RamPO effectively enhances the capability of the LLM to solve complex multi-hop questions within RAG settings. Notably, RamPO achieves these improvements without requiring additional human-annotated data, which is particularly beneficial in resource-constrained settings.

RamPO has a lower latency than tree-like inference methods but comparable performance. Although tree-like inference methods typically outperform chain-like inference methods, it suffers from high inference latency. This is attributed to the necessity of generating and evaluating multiple reasoning paths at each step, which substantially increases the number of generated tokens and, in turn, leads to higher computational cost and latency. In contrast, RamPO shifts the computational burden to the training phase, preserving the low-latency advantage of chain-like inference, which is 156.2× faster than tree-like inference methods on average, while offering comparable or superior performance. This demonstrates that RamPO can enhance performance without sacrificing efficiency.

5.4 The effectiveness of selection strategies of dispreferred nodes

We investigate the effect of various methods for selecting preference pairs on overall model performance. As illustrated in Figure 4, we experiment with four strategies based on the UCT of each node: 1) **Winners/ Losers**, in which all nodes on the reasoning paths with the correct answer are designated as preferred, while all remaining nodes are treated as dispreferred; 2) **Winners/ Lowest**, where for each parent node, only the child node with the lowest UCT leading to an incorrect answer is designated as dispreferred; 3) **Highest/ Losers**, where for each parent node, only the child node with the highest UCT leading to the correct answer is preferred; 4) **Highest/ Lowest**, wherein for each parent node, only the child node with the highest UCT leading to the correct answer is preferred and the child node with the lowest UCT leading to an incorrect answer is dispreferred. To ensure a fair comparison, we maintained an equal number of training questions across all strategies.

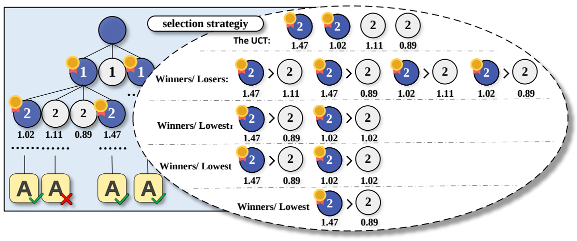


Figure 4: Different strategies for selecting dispreferred nodes.

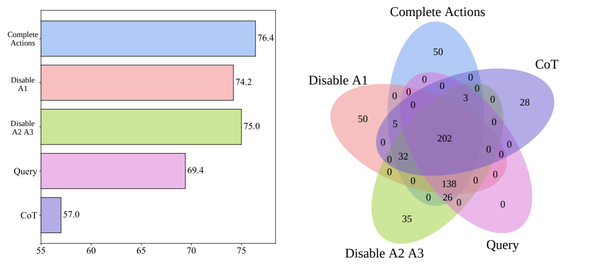
Strategy	Acc.(%)
Winners/ Lowest	36.5
Highest/ Losers	35.8
Highest/ Lowest	35.0
Winners/ Losers	38.7*

Table 3: Comparison between the selection strategies of dispreferred nodes on the model performance.

As shown in Table 3, the Winners/Losers strategy, which generates the most preference pairs, achieves the best performance, while the Highest/Lowest strategy with the least generated preference pairs performs the worst. This indicates that a coarse-grained distinction between preferred and dispreferred nodes is more effective than a finer-grained differentiation. The greater the number of preference pairs, the stronger the model's ability to distinguish preference relationships when choosing a reasoning path.

5.5 The effectiveness of actions on the Recall of Documents

To evaluate the effectiveness of action sequence in RamPO for document retrieval in RAG scenarios, we sequentially remove action A_1 and the combination of actions A_2 and A_3, and measure the recall rates of the gold-standard documents (supporting facts) on HotpotQA retrieved under five settings: 1) use the complete action sequence, 2) disable action A_1, 3) disable actions A_2 and A_3 together, 4) use intermediate steps from CoT, and 5) use the query-only.



(a) The recall rates of the supporting documents retrieved under five settings

(b) Venn diagram of the overlap for questions with all supporting documents recall

Figure 5: The Effectiveness of actions for the Recall of Documents on HotpotQA, using the LLaMA-3.2-3B-Ins as the base model.

Figure 5 shows that using the complete action sequence yields the best document recall performance. Removing actions leads to moderate performance drops, while both still outperform the rest settings baselines by a large margin. These findings suggest that the designed action sequence in RamPO plays a critical role in enabling recursive evidence aggregation, effectively guiding the retrieval process through a structured reasoning path. Each action contributes to progressively refining the query context and identifying relevant supporting documents. In particular, the full sequence facilitates a multi-hop reasoning mechanism, which allows the model to integrate and propagate information across multiple intermediate steps, enhancing its ability to recall the correct supporting facts.

5.6 The effectiveness of training data quantity

To assess the effect of the quantity of training data on the optimization of the model, we performed an ablation study that systematically varies the number of questions used to generate preference pairs, ranging from 0 to 200. As depicted in Figure 6, model performance initially deteriorates before showing improvement as the number of training questions increases. Specifically, when fewer instances are utilized, model performance falls below that of the untrained baseline, probably due to overfitting (Azar et al., 2024) and compromised generalization. However, with an increase to around 100 instances, performance consistently improves, eventually exceeding that of the base model. When the number of questions surpasses around 160, performance stabilizes, suggesting the model has reached a sufficient amount of data.

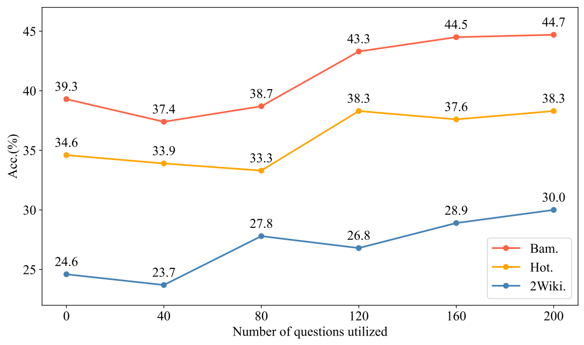


Figure 6: The effectiveness of the number of questions in model optimization.

5.7 The effectiveness of preference modeling

We investigate the effectiveness of preference modeling on model performance by conducting supervised fine-tuning (SFT) on the data without dispreferred counterparts. As shown in Table 4, the inclusion of dispreferred data enhances model performance. This observation implies that dispreferred nodes have a positive effect during optimization. It underscores the importance of utilizing both preferred and dispreferred steps to bolster the model's reasoning capabilities.

	DPO		SFT	
	Acc.(%)	Num./Ins.	Acc.(%)	Num./Ins.
Bam.	45.3	66	43.5	20
Hot.	38.3	86	33.9	27
2Wiki.	30.3	40	28.6	16

Table 4: The Effectiveness of preference modeling, where Num./Ins. refers to the average number of training data instances generated per question instance.

In addition, incorporating dispreferred nodes into preference pair construction can significantly enhance the efficiency of the training data construction. Compared to only using preferred nodes as training data, RamPO can generate an average of 43 more training instances per problem instance, representing a 204% increase, suggesting that RamPO requires only a small number of human annotated samples by design. Constructing the reasoning tree is a time-consuming process, RamPO offers a practical trade-off between efficiency and effectiveness, allowing for efficient resource management while still achieving significant improvements.

6 Conclusion

In this work, we introduced RamPO, a novel framework that integrates MCTS with preference-based optimization for multi-hop question answering in the RAG setting. By leveraging MCTS-guided heuristic exploration to constrain the search space and aligning with preference preferred by MCTS in offline, RamPO provides a trade-off between latency and reasoning accuacy during search space exploration in online inference.We provide empirical evidence that RamPO enhances the reliability and robustness of reasoning in LLMs by heuristically exploring and evaluating the search space. Extensive experiments across diverse LLMs and challenging multi-hop QA benchmarks validate the effectiveness of RamPO, showing significant gains in performance and substantial reductions in online inference latency.

Limitations

Although our proposed RamPO framework leverages MCTS to efficiently constrain the reasoning search space, enabling the model to generate high-quality reasoning chains during inference, there are still limitations in the current sample granularity. Specifically, we adopt a sequence-level sampling strategy, where the order of actions within a reasoning chain is fixed during training and inference. This design, while contributing to output stability, limits the model's flexibility in exploring diverse reasoning paths.

A potential direction for future work is to elevate the sampling granularity from the sequence level to the action level. This would allow the model to dynamically sample and select the next reasoning action at each step, guided by MCTS. Such an approach could encourage the model to learn to generate robust, high-quality reasoning chains while adapting its actions to different inference trajectories. This line of exploration may further enhance the model's generalizability and interpretability in complex reasoning tasks.

Ethical Considerations

It is a widely held view that large language models (LLMs) have the capacity to produce predictions that may be biased. This concern gains particular significance when the input queries contain attributes that are sensitive in nature. Given the potential for bias and other related issues, this study strongly recommends confining the use of such models to research contexts. Extra caution is imperative when considering the application of these models beyond research applications.

In this study, our use of existing artifacts is consistent with their intended purposes. All the datasets and models used in this work are publicly available. Specifically, LLaMA-3.2-3B-Instruct have Llama 3.2 Community License Agreement [1]. LLaMA-3.1-8B-Instruct have Llama 3.1 Community License Agreement [2]. Qwen-2.5-7B-Instruct, HotpotQA dataset, and 2WikiMultiHopQA (2Wiki) dataset have Apache-2.0 license [3]. Bamboogle dataset has MIT License [4].

[1] https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct/blob/main/LICENSE.txt

[2] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct/blob/main/LICENSE

[3] https://www.apache.org/licenses/LICENSE-2.0

[4] https://opensource.org/license/MIT

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. 2024. Rq-rag: Learning to refine queries for retrieval augmented generation. *arXiv preprint arXiv:2404.00610*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2023. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. 2025. Deeprag: Thinking to retrieval step by step for large language models. *arXiv preprint arXiv:2502.01142*.

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

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Yunhai Hu, Yilun Zhao, Chen Zhao, and Arman Cohan. 2025. Mcts-rag: Enhancing retrieval-augmented generation with monte carlo tree search. *arXiv preprint arXiv:2503.20757*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315, Singapore. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, and 1 others. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Lei Han, Haitao Mi, and Dong Yu. 2024. Toward self-improvement of llms via imagination, searching, and criticizing. *Advances in Neural Information Processing Systems*, 37:52723–52748.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022a. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models, 2023. *URL https://arxiv.org/abs/2210.03629*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772.

Jinghan Zhang, Xiting Wang, Weijieying Ren, Lu Jiang, Dongjie Wang, and Kunpeng Liu. 2024b. Ratt: A thought structure for coherent and correct llm reasoning. *Preprint*, arXiv:2406.02746.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. 2024c. Chain of preference optimization: Improving chain-of-thought reasoning in llms. *Advances in Neural Information Processing Systems*, 37:333–356.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. *Preprint*, arXiv:2205.10625.

A Implementation Appendix

In this work, we set the default configurations of RamPO as follows:

Parameter	**Default Value**
MCTS Configuration	
MCTS Exploration Weight	$\sqrt{2}$
Max Iteration	1000
Max Answer Candidates	40
Number of Samples	6
LLM Configuration	
Temperature	0.4
Top-K	50
Top-P	1.0

Table 5: Default configurations of RamPO.

In the experiments, we used the following datasets: Bamboogle [5], HotpotQA [6] and 2WikiMultiHopQA [7]. To maintain a reasonable budget, especially given the high computational demand of ToT, we limit each dataset to a maximum of 400 test samples through random sampling. The LLMs employed are LLaMA-3.2-3B-Instruct [8], LLaMA-3.1-8B-Instruct [9] and Qwen-2.5-7B-Instruct [10]. For all baseline methods, we set the model configurations identical to those in Table 5. In the process of MCTS, the exploration phase continues until either the number of answer candidates reaches 40 or the maximum number of iterations is achieved. For efficient fine-tuning, we use Low-Rank Adaptation (LoRA) adapters (Hu et al., 2022).The learning rates for DPO and SFT are 5e-6 and 1e-5 respectively. For LoRA, the rank is set to 8, and α is set to 16. for DPO, β is set to 0.2.

B Prompt Examples

We construct a unified set of prompts for the three datasets. For each action that requires generation by the LLM, some demonstrations are provided in the prompts. Specifically, the prompts for different actions are presented below.

[5] https://github.com/ofirpress/self-ask/tree/main

[6] https://hotpotqa.github.io

[7] https://github.com/Alab-NII/2wikimultihop

[8] https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

[9] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

[10] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct

A_1: Query rephrasing

Given an input question, rephrase it into a more intuitive and easier-to-understand version. The original question is enclosed within <Original Question> </Original Question> tags, and the corresponding rephrased question is enclosed within <Rephrased Question> </Rephrased Question> tags.
<Original Question>
In what school district is Governor John R. Rogers High School, named after John Rankin Rogers, located?
</Original Question>
<Rephrased Question>
What school district is Governor John R. Rogers High School in?
</Rephrased Question>
<Original Question>
Which Australian racing driver won the 44-lap race for the Red Bull Racing team?
</Original Question>
<Rephrased Question>
Who is the Australian racing driver that won a 44-lap race for the Red Bull Racing team?
</Rephrased Question>
<Original Question>
What star of *Parks and Recreation* appeared in November?
</Original Question>
<Rephrased Question>
Which actor from *Parks and Recreation* made an appearance in November?
</Rephrased Question>
<Original Question>
Which genus of flowering plant is found in an environment further south, Crocosmia or Cimicifuga?
</Original Question>
<Rephrased Question>
Between Crocosmia and Cimicifuga, which plant genus is typically found further south?
</Rephrased Question>
<Original Question>
In what year did the man who shot the Chris Stockley, of The Dingoes, die?
</Original Question>
<Rephrased Question>
What is the year of death for the man who shot Chris Stockley, of The Dingoes?
</Rephrased Question>
<Original Question>
{User Query}
</Original Question>
<Rephrased Question>

A_2 and A_3: Query decomposition and Subquestions solving

Given an input question, decompose it into indivisible sub-questions and answer them briefly. The original question will be enclosed in <Question> and </Question>.
<Question>
Who lived longer, Theodor Haecker or Harry Vaughan Watkins?
</Question>
<Subquestions_1>
When did Theodor Haecker die? Theodor Haecker was 65 years old when he died.
</Subquestions_1>
<Subquestions_2>
When did Harry Vaughan Watkins die? Harry Vaughan Watkins was 69 years old when he died.
</Subquestions_2>
<Question>
Why did the founder of Versus die?
</Question>
<Subquestions_1>
Who is the funder of Versus? The founder of Versus was Gianni Versace.
</Subquestions_1>
<Subquestions_2>
Why did Gianni Versace die? Gianni Versace was shot and killed on the steps of his Miami Beach mansion on July 15, 1997.
</Subquestions_2>
<Question>
Who is the grandchild of Dambar Shah?
</Question>
<Subquestions_1>
Who is the son of Dambar Shah? Dambar Shah (? - 1645) was the father of Krishna Shah.
</Subquestions_1>
<Subquestions_2>
Who is the son of Krishna Shah? Krishna Shah (? - 1661) was the father of Rudra Shah.
</Subquestions_2>
<Question>
{User Query}
</Question>
<Subquestions_1>

A_5: Document analysis

Based on the given question, we retrieve some documents. Please extract and summarize the content that is relevant to the question. If there is nothing relevant to the question in the document, answer "None".
Question: Butautas tried to depose his uncle who between which years?
Document: <doc>Marinus (praetorian prefect): Marinus was one of the most trusted and senior aides of the Byzantine emperor Anastasius I (r. 491–518). He served twice as praetorian prefect of the East, supervised some of Anastasius's tax reforms, supported the Emperor's pro-Monophysite policies and led the Byzantine navy in a crucial battle that ended for good the rebellion of general Vitalian in Thrace. He survived into the regime of Justin I (r. 518–527), when he held his second tenure as praetorian prefect, but was soon sidelined from power.</doc> <doc>Nuclear energy policy of the United States: The nuclear energy policy of the United States developed within two main periods, from 1954–1992 and 2005–2010. The first period saw the ongoing building of nuclear power plants, the enactment of numerous pieces of legislation such as the Energy Reorganization Act of 1974, and the implementation of countless policies which have guided the Nuclear Regulatory Commission and the Department of Energy in the regulation and growth of nuclear energy companies. This includes, but is not limited to, regulations of nuclear facilities, waste storage, decommissioning of weapons-grade materials, uranium mining, and funding for nuclear companies, along with an increase in power plant building. Both legislation and bureaucratic regulations of nuclear energy in the United States have been shaped by scientific research, private industries' wishes, and public opinion, which has shifted over time and as a result of different nuclear disasters.</doc> <doc>Butautas: Butautas (baptized "Henryk"; died on May 7, 1380 in Prague) was a son of Kęstutis, Grand Duke of Lithuania. He attempted to depose his uncle Algirdas and usurp power in Lithuania, but failed and was forced into exile. He joined the court of the Holy Roman Emperor and even inspired a poem about conversion to Christianity. Butautas is sometimes confused with his brother Vaidotas.</doc>
Summary: 1. Butautas, son of Kęstutis (Grand Duke of Lithuania), attempted to depose his uncle Algirdas and usurp power in Lithuania. 2. The attempt failed, leading to his exile. 3. The document does not specify the exact years of the attempted deposition. 4. Butautas died in 1380.
...{2 × demonstrations}...
Question:**{User Query}**
Document:**{Document}**
Summary:

A_6: Answer extraction

Task: Answer the given question step-by-step
Question: Who lived longer, Theodor Haecker or Harry Vaughan Watkins?
Step 1 Rephrased Question: Who had a longer lifespan, Theodor Haecker or Harry Vaughan Watkins?
Step 2 Subquestions_1: When did Theodor Haecker die? Theodor Haecker was 65 years old when he died.
Step 3 Subquestions_2: When did Harry Vaughan Watkins die? Harry Vaughan Watkins was 69 years old when he died.
Step 4 Document: <doc>In what year was Stephen Hawking born? 1942</doc> <doc>What is the birthdate of Ethan Hawke? November 6, 1970</doc> <doc>In what year was Ethan Hawke born? 1970</doc>
Step 5 So the final answer is: Harry Vaughan Watkins
Question: Why did the founder of Versus die?
Step 1 Rephrased Question: What caused the founder of Versus to pass away?
Step 2 Subquestions_1: Who is the funder of Versus? The founder of Versus was Gianni Versace.
Step 3 Subquestions_2: Why did Gianni Versace die? Gianni Versace was shot and killed on the steps of his Miami Beach mansion on July 15, 1997.
Step 4 Document: <doc>What is the birthdate of Avicii? September 8, 1989</doc> <doc>In what year was Avicii born? 1989</doc> <doc>What is the birthplace (country only) of Leonardo da Vinci? Italy</doc>
Step 5 So the final answer is: Shot
Question: What is the capital of the birthplace of Edin Dzeko?
Step 1 Rephrased Question: What is the main city of the place where Edin Dzeko was born?
Step 2 Subquestions_1: Where was Edin Dzeko born? Edin Dzeko was born in Bosnia and Herzegovina.
Step 3 Subquestions_2: What is the capital of Bosnia and Herzegovina? Sarajevo
Step 4 Document: <doc>What is the birthplace (country only) of Edin Dzeko? Bosnia And Herzegovina</doc> <doc>What is the capital of Bosnia and Herzegovina? Sarajevo</doc>
Step 5 So the final answer is: Sarajevo
Question: **{User Query}**