

# Everything of Thoughts : Defying the Law of Penrose Triangle for Thought Generation


Ruomeng Ding<sup>†\*</sup>, Chaoyun Zhang<sup>‡</sup>, Lu Wang<sup>‡</sup>, Yong Xu<sup>‡</sup>, Minghua Ma<sup>‡</sup>,  
Wei Zhang<sup>§</sup>, Si Qin<sup>‡</sup>, Saravan Rajmohan<sup>‡</sup>, Qingwei Lin<sup>‡</sup>, Dongmei Zhang<sup>‡</sup>

<sup>†</sup>Georgia Institute of Technology

<sup>‡</sup>Microsoft

<sup>§</sup>East China Normal University

## Abstract

This paper introduces a novel thought prompting approach called “Everything of Thoughts” (XoT) for Large Language Models (LLMs) to defy the law of “Penrose triangle ” of existing thought paradigms, to achieve three key perspectives in thought generation simultaneously: performance, efficiency, and flexibility. XoT leverages pretrained reinforcement learning and Monte Carlo Tree Search (MCTS) to incorporate external domain knowledge and planning capability into thoughts, thereby enhancing LLMs’ decision-making capabilities. Through the MCTS-LLM collaborative thought revision framework, XoT autonomously produces high-quality comprehensive cognitive mappings with minimal LLM interactions. Additionally, XoT empowers LLMs to utilize flexible cognitive mappings for solving problems with multiple solutions.

We evaluate XoT on several challenging problem-solving tasks, including Game of 24, 8-Puzzle, and Pocket Cube. Our results demonstrate that XoT significantly outperforms existing approaches in various dimensions, showcasing its remarkable proficiency in addressing complex problems across diverse domains. The data and code are available at <https://github.com/microsoft/Everything-of-Thoughts-XoT>.

## 1 Introduction

Recent advancements in Large Language Models (LLMs) have greatly advanced problem solving in diverse domains such as mathematical reasoning (Frieder et al., 2023), knowledge reasoning (Omar et al., 2023), root cause analysis (Chen et al., 2023) and causal inference (Kırcıman et al., 2023), etc.. This progress can be largely attributed to the technique of decomposing intricate problems into smaller language sequences referred to as “thoughts”. Through a step-by-step inference process involving the use of prompts, each thought functions as an intermediate stage, contributing to

\*Work done during an internship at Microsoft.

Table 1: Comparisons of different prompting paradigms.

Paradigm	Performance	Efficiency	Flexibility
IO	✗	✓	✗
CoT	✗	✓	✗
CoT-SC	✗	✗	✗
ToT	✓	✗	✗
GoT	✓	✗	✓
XoT	✓	✓	✓

the simplification of tackling complex problems to fulfill the problem’s ultimate objective.

Effective design of thought toward complex problem-solving, whether for humans or LLMs, should prioritize three crucial aspects, namely:

- **Performance.** Performance is the accuracy of the solution to a problem, including the precision of each thought at intermediate stages. This holds paramount importance for problem-solving.
- **Efficiency.** Efficiency relates to the number of LLM inference calls required to solve a single problem. Minimizing this is crucial due to the high cost associated with LLM inference.
- **Flexibility.** Flexibility in thought topology refers to the diverse thought structures that can be employed by LLMs for problem-solving. These may include chains, trees, or even graphs, mirroring human thought processes. Enabling more flexible thought structures enhances LLMs’ divergent and creative thinking capability, especially those with multiple potential solutions.

There exist several thought paradigms, such as Chain-of-Thought (CoT) (Wei et al., 2022), Tree-of-Thought (ToT) (Yao et al., 2023), and Graph-of-Thought (GoT) (Besta et al., 2023), etc.. However, these paradigms each have their limitations and cannot simultaneously achieve all the three desired attributes, as illustrated in Table 1. Specifically, direct Input-Output (IO) prompting is suitable primarily for simple problem-solving scenarios with single-step processes, lacking both in performance

and flexibility. CoT and self-consistency CoT (CoT-SC) enable step-by-step problem solving, resulting in modest performance improvements, but they are confined to linear thought structures, limiting their flexibility. In contrast, ToT and GoT permit more versatile thought topologies, accommodating tree-like or graph-like structures. However, they require the evaluation of intermediate thought steps through LLM itself, incurring significant computational costs due to multiple LLM calls. These paradigms are constrained by a law analogous to the “Penrose triangle  $\triangle$ ”, wherein they can achieve a maximum of two of the attributes, and none of them can simultaneously attain all three.

We propose a novel solution called “Everything of Thoughts” (XOT) to address the limitations of conventional thought frameworks, enhancing essential attributes of thought generation, including performance, efficiency, and flexibility for LLM inference. XOT leverages reinforcement learning (RL) (Li, 2017) and MCTS (Silver et al., 2017), in conjunction with lightweight policy and value networks, to pretrain on specific tasks for thought searching and generalize to new problems. This pretraining effectively integrates external domain knowledge and planning capability into the “thoughts” provided to LLMs, expanding their problem-solving capabilities, and thereby significantly improving **Performance**.

Once trained, XOT efficiently performs thought searching using MCTS with cost-effective policy and value networks for exploration and autonomously generates complete cognitive mappings and inject external knowledge to LLMs. It then employs a **MCTS-LLM collaborative thought revision process** to further improve the thought quality while minimizing LLM interactions, as will be illustrated in Sec. 3.4 and Fig. 3. This eliminates the need for LLMs to explore and evaluate thoughts themselves, as required by ToT and GoT, enhancing XOT’s **Efficiency**. Furthermore, MCTS demonstrates remarkable **Flexibility** as it can explore various thought topologies akin to those employed in human mind mapping processes (Faste and Lin, 2012; Jamieson, 2012). This enables diverse and creative thinking for LLMs, making it particularly valuable when dealing with complex thought structures or tasks featuring multiple potential solutions. By concurrently achieving superior performance, efficiency, and flexibility, XOT challenges the constraints posed by the “Penrose triangle  $\triangle$ ” law, significantly surpassing the

capabilities of other thought generation paradigms.

We evaluate XOT across three challenging problem-solving tasks, namely Game of 24, 8-Puzzle, and Pocket Cube. Our experimental results consistently show XOT’s superior performance, and its capacity to provide multiple solutions to problems efficiently with just a few LLM calls. These establish XOT as an effective thought generation approach, paving the way for new avenues in LLMs’ problem-solving capabilities.

## 2 Background

**Thought for LLMs.** Addressing complex problems often entails breaking down the overarching objective into multiple intermediary steps. The cognitive processes associated with each step are thoughts, which can be expressed as linguistic sequences for LLMs’ problem-solving. Structures of these thought may take various forms, including chains, trees, or graphs, depending on how the thoughts are organized towards a solution.

**Input-Output (IO) Prompting (Fig. 1 (a)).** The IO method is the most straightforward approach to instruct LLMs to address a problem without the provision of any intermediate thought processes.

**Chain-of-thought (CoT) (Wei et al., 2022) (Fig. 1 (b)).** CoT decomposes problem-solving into a sequential chain of thoughts, allowing LLMs to approach complex problems step by step.

**Self-consistency CoT (CoT-SC) (Wang et al., 2023) (Fig. 1 (c)).** CoT-SC utilizes multiple instances of the CoT to produce multiple outputs from LLMs. It selects the best results from these outputs, providing more robust and consistent inference than the standard CoT.

**Tree-of-thought (ToT) (Yao et al., 2023) (Fig. 1 (d)).** ToT organizes thoughts in a tree structure, using search algorithms for solution finding. However, it requires multiple LLM inference calls, making it costly and inefficient.

**Graph-of-thought (GoT) (Besta et al., 2023) (Fig. 1 (e)).** GoT extends ToT by allowing graph-like thought structures through thought consolidation during search phases. Despite its flexibility, it needs multiple LLM inference calls, leading to high computational costs.

## 3 XOT: Everything of Thoughts

XOT serves as an LLM-MCTS collaborative framework designed to enhance the thought generation process, thereby assisting LLMs in resolving com-

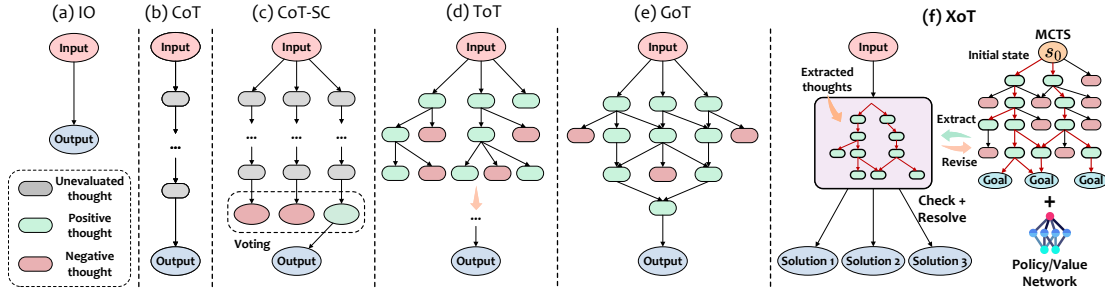


Figure 1: Comparison of XOT versus other prompting paradigms.

plex problems. It leverages MCTS for proficient and efficient thought exploration while harnessing the capabilities of LLMs to refine and amend the thoughts derived from MCTS. This synergistic interaction creates a mutually beneficial arrangement, ultimately enabling the successful resolution of intricate problems characterized by high levels of performance, efficiency, and flexibility.

### 3.1 XOT in a Nutshell

We present an overview of the architecture of XOT in Fig. 1 (f). XOT comprises two key components: (i) a MCTS module guided by policy/value networks; and (ii) an LLM solver for thought revision and inference.

During training, MCTS is harnessed to explore potential thought structures for a specific task through simulated scenarios. This process entails the recording of states, values, and the visitation frequencies of thought nodes in each simulation. These recorded data are subsequently employed to iteratively train the policy and value estimation model, enabling it to assimilate domain knowledge and comprehend the world model. Once trained, the estimated policy and value are utilized to guide the MCTS to search for a thought trajectory to aid LLMs in problem-solving. Note that these thoughts do not provide LLMs with definitive or error-free answers, as they may contain inaccuracies or suboptimal solutions. LLMs are responsible for reviewing and refining these thoughts when they seem erroneous or require adjustments. They continue MCTS the search process if needed and eventually formulate the final answers by integrating these external thoughts with their internal knowledge.

### 3.2 Thought Searching Formulation

The fundamental objective of employing the thought generation paradigm for LLMs is to identify the optimal decomposition of a complex problem into several manageable sub-steps. Each sub-step aims to alter the current status of the problem,

eventually culminating in the successful resolution of the overarching problem. This approach, as seen in ToT and GoT, hinges on well-defined state transitions and clear final objectives. Consequently, it is natural to conceptualize the thought-searching process as a Markov Decision Process (MDP) (Puterman, 1990), in which:

- **State**  $s_t$ : Represents the current status of the problem. The initial state  $s_0$  corresponds to the original problem, while intermediate states are characterized by either decomposed sub-problems or the results of their resolution.
- **Action**  $a_t$ : Signifies the one-step solution or action associated with tackling a problem, leading to a transition to a new state.
- **Reward**  $r$ : Reflects the comprehensive evaluation of the solution to the original problem, assessing whether it has been effectively resolved through the process of problem decomposition.
- **Thought**  $\tau$ : A one-step thought is a combination of one-step state and action, *i.e.*,  $\tau = \{s, a\}$ . This formulation encapsulates the process of decomposing a complex problem into multiple sub-tasks and their respective outcomes.

The detailed definitions of state, action, reward and thought for each task are shown in Table 9, Appendix A. The generation of complete thoughts  $\mathcal{T} = \{\tau_1, \dots, \tau_N\}$ , can be construed as the endeavor to discover a thought trajectory to maximize the accumulated reward to address the problem.

### 3.3 Thoughts Searching with MCTS

The formulation above naturally aligns the thought within LLM as a state-action pair. This approach facilitates the effective exploration of its optimal trajectory using a combination of MCTS and RL. This adheres to an iterative simulation cycle that encompasses three key phases: selection, expansion & evaluation, and backpropagation. It heavily

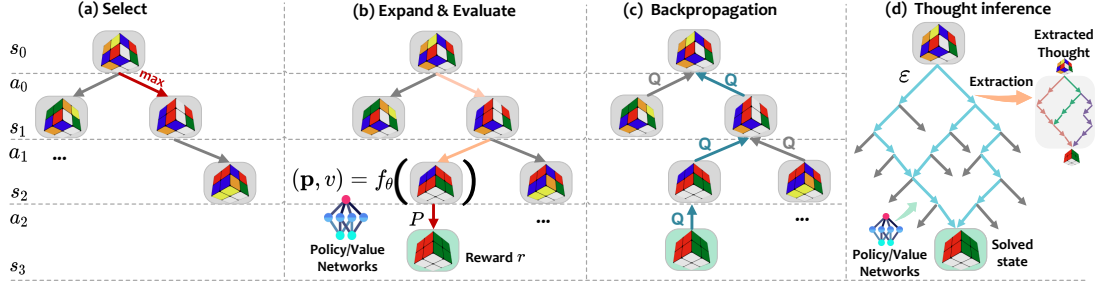


Figure 2: An illustration of phases in MCTS for thought searching ((a)-(c)) and thought inference (d).

depends on the utilization of neural networks  $f_\theta$ , which simultaneously estimate the value and action probability for a given state  $s_t$ . The aim is to reduce the number of rollouts and accelerate the search process, similar to the approach employed in AlphaGo Zero (Silver et al., 2017). We provide a visual representation of an iteration of the MCTS in Fig. 2 (a)-(c) by taking Pocket Cube as an example and detail each process below.

**Selection.** In the selection phase, the algorithm initiates at the root node and proceeds to choose an action  $a^*$  from the available set  $\mathcal{A}(s)$  for single-step thought generation in the current state  $s$ . This process continues until a leaf node within the current tree is reached. The selection is guided by the PUCT algorithm (Rosin, 2011), aiming to maximize the Upper Confidence Bound (UCB) (Garivier and Moulines, 2011), as follows:

$$a^* = \arg \max_{a \in \mathcal{A}(s)} \left[ Q(s, a) + w \cdot P_\theta(s, a) \sqrt{\frac{N(s)}{1+N(s, a)}} \right]. \quad (1)$$

Here,  $Q(s, a)$  denotes the Q-value of a state-action pair  $(s, a)$ , which estimates the quality of a particular action in a given state. The higher the Q-value, the better the action is considered to be.  $P_\theta(s, a)$  denotes the predicted prior probability of selecting action  $a$  given the state  $s$  obtained from a neural network  $f_\theta$ , and  $N(s, a)$  represents the count of times action  $a$  has been chosen in state  $s$ . The parameter  $w$  controls the trade-off between exploration and exploitation. The selection process will continue until an unexplored node is encountered.

**Evaluation and Expansion.** Upon reaching a previously unselected leaf node, we expand to the state  $s$  for the next step for new thought exploration. This expansion involves the evaluation of its value and action probability on the state, which are modeled by neural networks parameterized by  $\theta$ , i.e.,  $(P_\theta(s), v_\theta(s)) = f_\theta(s)$ . Here  $P_\theta(s)$  is the prior probabilities for all actions on  $s$ , and  $v_\theta(s)$

denotes its predicted state value. These two values are retained and stored for backup purposes, and state  $s$  is masked as “visited”.

**Backpropagation.** Following the expansion of a leaf node in the above phases, which could be either an unexplored or terminal state, the algorithm proceeds to update all the  $Q(s, a)$  values via backpropagation. For unexplored nodes, this update involves computing the mean of its estimated value  $v_\theta$ , while for terminated nodes, it’s based on the true reward  $r$ . These updates occur as information is backpropagated along the trajectory to subsequent nodes. The visit count for each state-action pair is also incremented as follows:  $N(s, a) = N(s, a) + 1$ .

A simulation is completed after a sequence of selection, evaluation, expansion, and backpropagation steps. After conducting multiple simulations, we proceed to the next step by selecting an action at state  $s$  using a probability distribution defined as  $\varepsilon_a \propto N(s, a)^{1/\gamma}$ , where  $\gamma$  is a temperature constant that regulates the level of exploration.

**Policy and Value Networks Training.** The simulations described above allow us to compile a dataset for each sample state  $s$  containing  $(s, \varepsilon(s), v(s))$ , where  $\varepsilon(s) = \{\varepsilon_a \mid a \in \mathcal{A}(s)\}$ , and  $v(s)$  represents the ground truth value obtained by accumulating rewards along the trajectory starting from state  $s$ . Subsequently, we can train a combined policy and value network  $f_\theta$  to minimize the discrepancy between the predicted value  $v_\theta(s)$  and the actual value  $v(s)$ , while also maximizing the alignment between the action probabilities produced by the neural network  $P_\theta(s)$  and the search probabilities  $\varepsilon(s)$ . This can be achieved by minimizing the following loss function:

$$\mathcal{L} = (v(s) - v_\theta(s))^2 + \varepsilon(s)^T \log P_\theta(s). \quad (2)$$

This training iterates alongside the simulation process to continually enhance the performance of  $f_\theta$ , resulting in progressive improvements in thought searching capabilities.



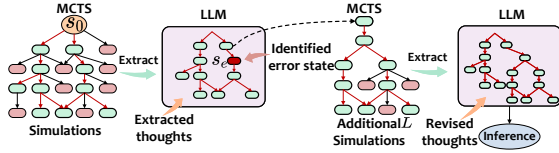


Figure 3: The thought revision process in XOT.

### 3.4 Thought Inference with MCTS

Once trained, we utilize the  $f_\theta$  to guide the MCTS in generating a thought for a new problem, which assists the LLM in solving it. Specifically, MCTS is utilized to perform  $K$  simulations aimed at thought searching and problem-solving, as illustrated in Fig.2 (d). In each simulation,  $f_\theta$  is employed to guide the MCTS in its search for a thought trajectory. Throughout the training process,  $f_\theta$  incorporates external information related to the state and action quality. This information helps LLMs understand the world model, enhancing their long-term reasoning and planning abilities, which are areas they may not excel in (Stechly et al., 2023; Valmeekam et al., 2023), thereby ensuring the performance of thought generation. Once the simulation concludes, we record the visiting count  $N(s, a)$  and the thought trajectory is obtained based on the number of solutions required:

- **Single solution.** Starting from  $s$ , the action with the highest visiting count  $N(s, a)$  is selected.
- **Multiple solution.** We sample  $M$  thought trajectories following the probability distribution  $\varepsilon_a \propto N(s, a)$  and remove duplicates.

This results in one or multiple thought trajectories  $\mathcal{T}^*$  that consist of a sequence of state-action pairs for problem-solving. The trajectories for multi-solution problems may intertwine and converge at the same goal state, resulting in a graph-like thought structure. This demonstrates that XOT is capable of generating thought structures with flexibility. These trajectories are then transformed into text sequences that are concatenated to form a new prompt sequence provided to LLMs, even in the case of problems with multiple solutions. Therefore, we only require a single LLM inference call at this stage. Given that the  $f_\theta$  network is lightweight, this ensures the efficiency of XOT.

**Thought-to-Prompt Parsing.** Once the thought trajectories  $\mathcal{T}^*$  are extracted from MCTS, we convert them into a textual format necessary for LLM inference. In this conversion process, we transform both the state and action at each step of the thought,

*i.e.*,  $\tau = \{s, a\}$  in  $\mathcal{T}^*$ , into text. This conversion aims to provide a comprehensive state transition, facilitating LLMs in better understanding the task step by step. In the case of multi-solution scenarios, multiple trajectories are concatenated. This format remains consistent across all baselines, and the resulting prompt text is then fed to LLMs for inference or thought revision.

**Thought Revision.** Note that that MCTS may not always provide the globally optimal thought trajectory to directly solve the problem flawlessly. Therefore, the thoughts extracted from MCTS serve as a reference thinking process for the problem, aiding LLMs in a supportive capacity. The LLMs will leverage their internal knowledge to review the extracted thought, identify errors in the thought trajectory, and then ground its knowledge in collaboration with the MCTS to revise the thought. In this context, LLM plays a role akin to a participant to guide MCTS to enhance its performance.

The revision process is iterative in nature, as shown in Fig. 3. Initially, upon obtaining the extracted thought, we instruct the LLM to detect any errors in the thought generated by MCTS using its internal knowledge. If the LLM identifies an error, it results in an error state denoted as  $s_e$  within the thought. If no error is found, the thought remains unchanged. Starting from the parent state of  $s_e$ , MCTS conducts an additional  $L$  simulations, ultimately yielding a revised thought for the LLM. In scenarios involving multiple solutions, each solution undergoes this process individually. Upon the completion of the revision, we supply the LLMs with the revised thoughts for problem-solving. The revision process can be repeated several times to enhance the reliability of the answer.

Since LLMs are solely utilized for identifying errors during the revision process with only one call, the efficiency of XOT is maintained. The collaborative revision framework harnesses the strengths of both MCTS and LLMs. The thoughts generated by MCTS serve as reference or external knowledge, expanding the capabilities of LLMs into new domains. Simultaneously, LLMs leverage their internal knowledge to revise the extracted thoughts from MCTS. This takes effect as generating the entire thought path challenging, while identifying mistakes aligns better with the capabilities of LLMs. This dynamic interaction positions the LLM as an active participant in the game-solving process, guiding MCTS towards improved performance.

## 4 Experiment

**Experiment Setting.** We conduct an extensive evaluation of our XOT approach in comparison to several baseline methods across three challenging tasks: the Game of 24, the 8-Puzzle (with a  $3 \times 3$  grid), and the  $2 \times 2$  Pocket Cube. Detailed illustration of these tasks is provided in Appendix A. These tasks are characterized by their complexity, requiring multiple steps for completion and potentially having multiple solutions. To assess the effectiveness of XOT, we compare it against IO, CoT, CoT-SC, ToT, GoT, and single MCTS without LLMs for inference and revision. We also finetune LLaMA-2-13B (Touvron et al., 2023) for comparison, using the same training data and ground truth labels. We employ both GPT-3.5 (Ouyang et al., 2022) and GPT-4 (OpenAI, 2023) for these evaluations. The setup of XOT and baselines can be found in Appendix B, and the configuration of the Policy / Value Network used is provided in Appendix C.

**Evaluation Metric.** For each task, we assess the accuracy of each approach on the test set. Additionally, we track the number of LLM invocations required for all approaches to solve a problem, as well as the number of times  $f_\theta$  is invoked in the case of XOT. Note that  $f_\theta$  is a considerably smaller model compared to LLMs. In the context of multi-solution scenarios, we employ Multi-solution Accuracy (**MultiAcc.**) calculated as the average percentage of correctness across all solutions offered. Furthermore, we capture the total count of distinct solutions provided by each approach, regardless of their correctness, represented as **#Sol**. Note that we set the maximum solution number to 3 for all problems in multi-solution scenarios. All results are obtained from a single run. In Table 2 to Table 7, the number of thought **revision** is denoted by **r**.

### 4.1 Game of 24

The Game of 24 presents an arithmetic challenge wherein the goal is to employ four numbers within the range of 1 to 13, with basic arithmetic operations, (*i.e.*,  $+$ ,  $-$ ,  $\times$ ,  $\div$ ), to attain a final result of 24. This game may possess multiple valid solutions.

#### 4.1.1 Results

Table 2 displays the overall performance of all methods on this task. Notably, XOT consistently outperforms other baselines on both GPT-3.5 and GPT-4, achieving an accuracy of 79.56% and 74.45% respectively, with 1-time revision. However, after 3-time revision process, XOT’s accuracy

Table 2: Performance comparison on Game of 24.

Model	GPT-3.5			GPT-4		
	Acc. [%]	LLM invoked	$f_\theta$ invoked	Acc. [%]	LLM invoked	$f_\theta$ invoked
IO	6.57	1.00	-	10.22	1.00	-
CoT	2.19	1.00	-	4.38	1.00	-
CoT-SC	2.19	10.00	-	4.38	10.00	-
ToT (b=1)	5.84	22.11	-	34.31	23.50	-
ToT (b=3)	10.22	43.96	-	60.58	39.83	-
GoT (k=1)	2.92	7.00	-	10.95	7.00	-
LLaMA-2-13B	2.19	-	-	2.19	-	-
MCTS	62.77	-	-	62.77	-	-
<b>XoT (w/ 1 r)</b>	<b>79.56</b>	<b>1.39</b>	<b>92.15</b>	<b>74.45</b>	<b>1.38</b>	<b>88.20</b>
<b>XoT (w/ 2 r)</b>	<b>88.32</b>	<b>1.58</b>	<b>93.87</b>	<b>83.94</b>	<b>1.57</b>	<b>89.63</b>
<b>XoT (w/ 3 r)</b>	<b>90.51</b>	<b>1.72</b>	<b>95.94</b>	<b>85.40</b>	<b>1.78</b>	<b>92.48</b>

Table 3: Performance comparison on Game of 24 in the multi-solution scenario.

Model	GPT-3.5				GPT-4			
	Multi Acc.	#Sol	LLM invoked	$f_\theta$ invoked	Multi Acc.	#Sol	LLM invoked	$f_\theta$ invoked
IO	4.87	2.88	1.00	-	8.27	2.99	1.00	-
CoT	1.22	2.77	1.00	-	7.79	2.94	1.00	-
CoT-SC	1.70	2.76	10.00	-	8.03	2.99	10.00	-
ToT (b=3)	3.41	2.99	43.96	-	39.90	2.78	39.83	-
GoT (k=3)	8.03	1.93	7.00	-	10.46	1.39	7.00	-
<b>XoT (w/ 1 r)</b>	<b>62.90</b>	<b>2.29</b>	<b>3.51</b>	<b>116.34</b>	<b>76.25</b>	<b>2.36</b>	<b>2.31</b>	<b>109.64</b>

substantially improves to 90.51% and 85.40% for GPT-3.5 and GPT-4 respectively. This underscores the impressive performance of XOT, and demonstrates that the revision process significantly enhances performance, with only a limited increase in the utilization of LLM and  $f_\theta$ . Interestingly, the revision process in XOT mitigates the performance gap attributable to the modeling ability in this task. As we observe that XOT with GPT-3.5 achieves higher accuracy after revision compared to GPT-4.

Moreover, XOT consistently outperforms the use of MCTS solely. The performance advantages exhibit growth with the number of revision iterations, underscoring the complementary roles of LLM and MCTS, emphasizing their joint necessity in achieving superior results. The fine-tuned LLaMA-2-13B is only successful on 2.19% of the test data. This performance is lower than the IO method, indicating that the finetuning method is not suitable for planning tasks like the Game of 24. The best-performing prompting baseline, ToT (b=3) on GPT-4, attains an accuracy of 60.58%. However, it demands a substantial number of LLM invocations (39.83), which results in inefficiency. In contrast, XOT only requires less than 1.8 calls with revision. Although XOT requires some inference calls for  $f_\theta$ , the model is significantly less complex than LLM, making it a much more efficient approach.

Table 3 presents the performance of different methods in the multi-solution scenario. Over-

all, XoT remains the best-performing approach in terms of MultiAcc, significantly outperforming other baselines. Although XoT does not generate the most number of answers compared to other baselines, it generates more accurate answers, as its MultiAcc significantly outperforms other approaches. Notably, generating multiple solutions does not significantly increase XoT’s complexity, as it only requires 2.31 LLM calls with GPT-4 and around 100 calls for a smaller  $f_\theta$ , making it remain efficient. Overall, the remarkable performance of XoT in the multi-solution scenario demonstrates its ability to generate complex thoughts.

## 4.2 8-Puzzle

The 8-Puzzle is a classic sliding puzzle game that consists of a  $3 \times 3$  grid with eight numbered tiles and one empty space. Its objective is to rearrange the tiles from an initial configuration into a target state. This task may also have multiple solutions.

### 4.2.1 Results

The inherent spatial complexity of the 8-Puzzle, the need for long-term planning, and the presence of invalid actions create a significant challenge for LLMs, which rely solely on textual data as input. This challenge is starkly evident in the poor performance of the baselines on both GPT-3.5, where its IO prompting achieve a mere 0% success rate. XoT successfully addresses this issue by supplying thoughts acquired from MCTS, thereby infusing external knowledge into the problem-solving process. This augmentation empowers LLMs to tackle problems that were previously insurmountable. In summary, when using GPT-4, XoT achieves an accuracy of 93.28% with 1 revision and 95.80% with 3 revisions in the 8-Puzzle task, outperforming the best prompting baseline, ToT (b=3), which only achieves 13.45% accuracy. Additionally, XoT demonstrates efficiency, as it only requires approximately 1.6 LLM calls for 3-time revision setting. The poor performance of finetuned LLaMA-2-13B (0%) revealed a significant issue with hallucination. This underscores the inefficiency and ineffectiveness of finetuning approaches for tasks necessitating long-term planning, while also bringing to light the heightened costs associated with its use.

The multi-solution performance presented in Table 5 confirms that the XoT method continues to outperform other baselines for both GPT-3.5 and GPT-4 models in terms of MultiAcc, whether or not revision is applied. The revision process of

Table 4: Performance comparison on 8-Puzzle.

Model	GPT-3.5			GPT-4		
	Acc. [%]	LLM invoked	$f_\theta$ invoked	Acc. [%]	LLM invoked	$f_\theta$ invoked
IO	0.00	1.00	-	1.68	1.00	-
CoT	0.00	1.00	-	7.56	1.00	-
CoT-SC	0.84	10.00	-	8.40	10.00	-
ToT (b=1)	5.88	31.76	-	3.36	27.49	-
ToT (b=3)	6.72	55.86	-	13.45	54.13	-
GoT (k=1)	3.36	19.00	-	3.36	19.00	-
LLaMA-2-13B	0.00	-	-	0.00	-	-
MCTS	51.26	-	-	51.26	-	-
<b>XoT (w/ 1 r)</b>	<b>59.66</b>	<b>1.50</b>	<b>41.09</b>	<b>93.28</b>	<b>1.48</b>	<b>55.66</b>
<b>XoT (w/ 2 r)</b>	<b>59.66</b>	<b>1.92</b>	<b>42.18</b>	<b>94.96</b>	<b>1.55</b>	<b>58.91</b>
<b>XoT (w/ 3 r)</b>	<b>63.03</b>	<b>2.29</b>	<b>42.60</b>	<b>95.80</b>	<b>1.61</b>	<b>62.22</b>

Table 5: Performance comparison on 8-Puzzle in the multi-solution scenario.

Model	GPT-3.5				GPT-4			
	Multi Acc.	#Sol	LLM invoked	$f_\theta$ invoked	Multi Acc.	#Sol	LLM invoked	$f_\theta$ invoked
IO	0.00	2.47	1.00	-	0.84	2.97	1.00	-
CoT	1.43	2.05	1.00	-	7.84	1.21	1.00	-
CoT-SC	1.54	1.90	10.00	-	6.58	2.08	10.00	-
ToT (b=3)	2.52	2.98	55.86	-	5.60	2.97	54.13	-
GoT (k=3)	3.36	2.96	24.18	-	16.61	2.70	22.76	-
<b>XoT (w/ 1 r)</b>	<b>27.45</b>	<b>2.85</b>	<b>4.19</b>	<b>52.06</b>	<b>76.33</b>	<b>1.52</b>	<b>4.30</b>	<b>66.66</b>

XoT is particularly beneficial for GPT-4, as it improves the MultiAcc from 51.26% to 76.33%, compared to single MCTS. These results again demonstrate that XoT can effectively generate complex thought structures for multi-solutions with high performance and efficiency, making it particularly suitable for this task.

## 4.3 Pocket Cube

The  $2 \times 2$  Pocket Cube is a simplified variant of the classic Rubik’s Cube puzzle. Its objective is to restore all of its faces to a uniform color by executing rotations. This task may possess multiple solutions is known to be challenging to LLMs (cub).

### 4.3.1 Results

The Pocket Cube task, similar to the 8-Puzzle, poses a challenge that demands spatial imagination skills, making it difficult for LLMs to excel. As expected, most of the baselines show very poor performance in this task, with some baselines achieving 0% accuracy. The best prompting baseline, ToT (b=3) with GPT-4, only attains a success rate of 19.57%. In contrast, XoT can achieve over 77.60% accuracy with 1-time revision and over 80% accuracy with 3-time revision, establishing itself as an expert in solving this task. This is attributed to the injection of external knowledge from MCTS, enabling LLMs to solve problems that they would struggle with on their own. On the other hand, XoT improves accuracy by 30% compared to a

Table 6: Performance comparison on Pocket Cube.

Model	GPT-3.5			GPT-4		
	Acc. [%]	LLM invoked	$f_\theta$ invoked	Acc. [%]	LLM invoked	$f_\theta$ invoked
IO	1.09	1.00	-	1.09	1.00	-
CoT	0.00	1.00	-	1.09	1.00	-
CoT-SC	0.00	10.00	-	1.09	10.00	-
ToT (b=1)	7.65	16.50	-	11.48	16.39	-
ToT (b=3)	17.49	58.72	-	19.57	56.58	-
GoT (k=1)	1.64	8.93	-	18.03	8.55	-
LLaMA-2-13B	0.00	-	-	0.00	-	-
MCTS	46.44	-	-	46.44	-	-
XoT (w/ 1 r)	<b>74.32</b>	<b>1.55</b>	<b>64.63</b>	<b>77.60</b>	<b>1.54</b>	<b>75.51</b>
XoT (w/ 2 r)	<b>80.33</b>	<b>1.81</b>	<b>96.46</b>	<b>79.32</b>	<b>1.79</b>	<b>146.52</b>
XoT (w/ 3 r)	<b>84.70</b>	<b>2.01</b>	<b>103.22</b>	<b>83.61</b>	<b>2.00</b>	<b>84.63</b>

Table 7: Performance comparison on Pocket Cube in the multi-solution scenario.

Model	GPT-3.5				GPT-4			
	Multi Acc.	#Sol	LLM invoked	$f_\theta$ invoked	Multi Acc.	#Sol	LLM invoked	$f_\theta$ invoked
IO	0.27	2.00	1.00	-	1.09	1.98	1.00	-
CoT	0.55	1.05	1.00	-	0.82	1.91	1.00	-
CoT-SC	0.18	2.90	10.00	-	0.82	2.92	1.00	-
ToT (b=3)	5.83	2.99	58.72	-	6.52	2.99	56.58	-
GoT (k=3)	1.09	2.99	14.76	-	16.85	2.77	13.36	-
XoT (w/ 1 r)	<b>48.72</b>	<b>2.20</b>	<b>4.13</b>	<b>115.73</b>	<b>77.41</b>	<b>1.72</b>	<b>4.08</b>	<b>122.54</b>

single MCTS with one-time revision. This demonstrates the effectiveness of integrating MCTS and LLMs. Notably, XoT maintains high efficiency in this task, requiring only approximately 2 LLM inference calls for both GPT-3.5 and GPT-4. Again, the finetuned LLaMA-2-13B struggles with the Pocket Cube task (0%), due to significant hallucination issues. This comparison further validates the potential of XoT in contexts demanding extensive planning and decision-making accuracy.

In the case of the multi-solution scenario, the performance of the XoT method remains remarkable, achieving over 77% MultiAcc with GPT-4. The revision process continues to play an important role, significantly improving the performance of XoT with both GPT models. The closest competitor in this setting is GoT (k=3) with GPT-4, which achieves a MultiAcc of 16.85%, but it requires a significantly higher number of LLM invocations compared to XoT (13.36 vs. 4.08) and much lower MultiAcc. Overall, XoT retains its position as the best solution for the Pocket Cube.

## 5 Computational Training Costs of MCTS

The number of training and testing policy/value model calls for XoT are listed in Table 8. We train this model through three iterations, each comprising 10 self-play episodes for MCTS. Offline pre-training serves as a one-time solution that reduces

Table 8: Number of policy/value model calls in training and testing per iteration for different tasks.

	Game of 24	8-Puzzle	Pocket Cube
Training	1044.70	834.70	787.00
Testing	88.20	55.66	75.51

the computational burden of testing by integrating external knowledge. Methods like ToT and GoT, which rely solely on the LLMs’ internal knowledge, do not require pretraining but necessitate frequent calls to LLM during testing. For example, the average number of LLM invocations for three tasks in ToT are 39.83, 54.13, and 56.58, averaging 50.18 times per test problem. The computational cost of these recurring calls during testing exceeds the pretraining cost of the policy/value model in XoT.

Furthermore, it’s worth highlighting that GPT-3.5 boasts 175 billion parameters, and GPT-4 is estimated to have an astonishing over 1 trillion parameters. In contrast, the total number of parameters in the Policy/Value Network for all three tasks is approximately  $1e6$ . This deliberate design choice results in a model significantly smaller than LLMs, ensuring efficiency even with additional calls during training.

## 6 Case Study in Multi-Solution Scenarios

In Fig. 4, we provide examples of thought structures generated by XoT for all three tasks in the multi-solution scenario. Owing to the multiple solutions required, the generated thoughts intertwine during intermediate steps and converge towards the final goal state. This results in a naturally woven thought structure resembling a graph, showcasing the remarkable flexibility achieved by XoT. Upon closer examination of each example, in the case of the Game of 24, there are multiple solutions to reach the goal of 24. XoT effectively predicts these trajectories, indicating its ability to grasp complex thought structures. In the 8-Puzzle example, we observe instances of reflection in the thought structure, with back-and-forth recurrent state transitions. This demonstrates XoT’s capacity for self-reflection, a crucial attribute for LLMs, as discussed in (Shinn et al., 2023). In the case of the Pocket Cube, XoT identifies four distinct pathways to reach the goal state, leading to successful problem-solving across multiple solutions.

Overall, these cases highlight how XoT encapsulates the flexibility required in thought generation,



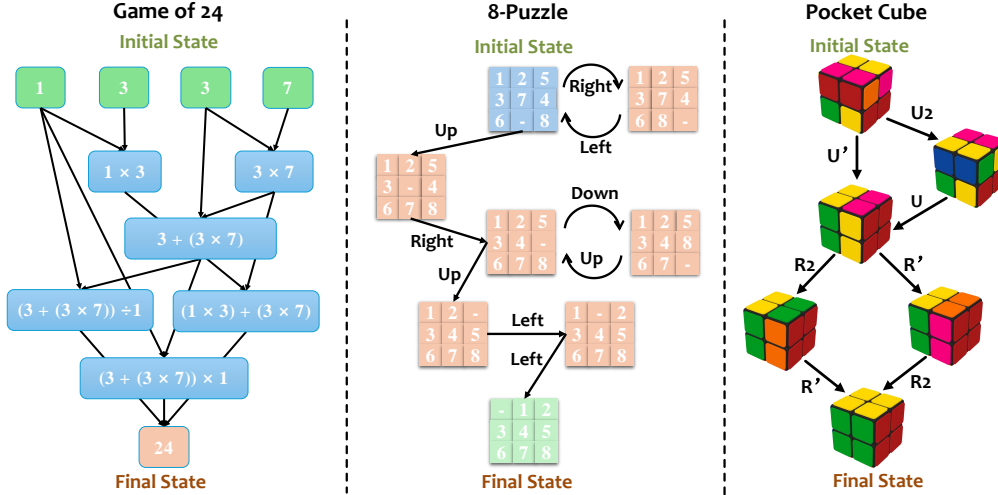


Figure 4: Examples of thought structures generated by XOT for all three tasks in the multi-solution scenario.

fostering diverse and creative thinking for LLMs. This enables them to produce multiple high-quality answers to a single problem effectively.

**Supplementary Insights.** We further conduct ablation study to assess the impact of thought revisions, the revision success rate, and the sensitivity to the completeness of the provided thoughts, presented in Appendix D.

## 7 Related Work

**Decision Making & Planning with LLMs.** The utilization of LLMs for decision-making and planning has become a prominent area of research. Similar to human problem-solving, the process involves breaking down complex problems into sub-tasks. Various frameworks, such as CoT (Wei et al., 2022), ToT (Yao et al., 2023), and GoT (Besta et al., 2023), have been designed to facilitate problem decomposition in different structural forms, leading to enhanced solutions derived from LLMs. Extensions of these frameworks have also been explored across different domains and modalities (Zhang et al., 2022, 2023; Ning et al., 2023; Turpin et al., 2023; Long, 2023). Our approach XOT distinguishes itself from the aforementioned work by concurrently achieving superior performance, efficiency, and flexibility, embodying the concept of comprehensive thought generation.

**Augmenting LLMs with MCTS.** MCTS is integrated with LLMs to enhance both training and inference processes. Hao et al., propose “Reasoning via Planning”, utilizing LLMs as a world model and reasoning agent, while combining MCTS as

a strategic explorer to enhance LLMs’ reasoning and planning abilities (Hao et al., 2023). Liu et al., incorporate MCTS and PPO (Schulman et al., 2017) to devise a value-guided decoding algorithm, thereby enhancing the preferability of generated text by LLMs (Liu et al., 2023). Additionally, Feng et al., employ MCTS to augment LLMs’ decoding and, consequently, their reasoning and planning capabilities (Feng et al., 2023). These studies underscore the significant potential of integrating MCTS with LLMs to improve their overall capabilities.

## 8 Conclusion

The XOT framework presented in this paper signifies a significant progression in thought generation for LLMs aimed at solving complex tasks. It challenges the constraints of the “Penrose Triangle  $\triangle$ ” by concurrently achieving performance, efficiency, and flexibility, a feat unattainable by existing prompting paradigms. This is achieved through the integration of MCTS with pretrained low-cost policy and value networks, by injecting domain knowledge and planning capability into LLMs, offloading thought searching, and facilitating unconstrained free-style thought exploration. The collaborative thought revision framework involving MCTS and LLM further enhances the quality of thought generation. Experimental evaluations conducted across three real-world problems, namely the Game of 24, 8-Puzzle, and Pocket Cube, show that our XOT framework significantly outperforms existing prompting paradigms, particularly in scenarios involving multi-solution problems.

## 9 Limitations

**Generalization to other tasks.** While XoT is presently utilized for reasoning and search problems capable of garnering rewards from real-world scenarios, its applicability can be extended to a broader spectrum of problem domains characterized by decomposable tasks with well-defined objectives. The main challenge in expanding the application of XoT to a broader range of tasks is the design of rewards, especially when explicit rewards are not directly obtainable from real-world environments. To address this, some studies have turned to using LLMs to get reward. The strategy of using LLMs for reward design is becoming increasingly popular and is a topic of ongoing research (Kwon et al., 2023). Therefore, it’s important to note that, in addition to the tasks employed in this paper, many other tasks can be formulated as MCTS searching problems, using LLMs to get rewards and rendering XoT applicable to a broader range of scenarios.

**Additional Training Costs.** We also note that the implementation of XoT necessitates the training of additional policy and value models to expedite the inference process. This training process requires the acquisition of datasets from real-world environments, introducing supplementary costs and efforts. However, note that these policy and value models are considerably smaller and more computationally efficient than the underlying LLMs, as discussed in Section 5. Consequently, the incurred costs are deemed low, particularly in the context of tasks featured in this study, where the thought steps and objectives are well-defined. In future research endeavors, we intend to explore methods to enhance the efficiency of the training process for XoT in scenarios where the objectives are less straightforward, such as multi-agent planning and code generation tasks (Talebirad and Nadiri, 2023; Vaithilingam et al., 2022). This endeavor will expand the applicability of the proposed XoT framework to a broader range of applications.

**Potential Risks.** XoT is susceptible to the MCTS module providing incorrect intermediate thoughts, which may result in an inaccurate final answer or hallucination. Changes in the environment could lead to inaccuracies in MCTS and subsequently in the thoughts provided to LLMs. However, LLMs have proven effective in revising thoughts by leveraging their internal knowledge, mitigating the risk associated with inaccuracies in the initial thought

generation. Additionally, LLMs may make mistakes and sometimes deviate from the thoughts generated by the MCTS module, leading to errors. This aspect should be taken into consideration when employing the approach.

## References

- 4 Numbers. <https://www.4nums.com/game/difficulties/>. [Online; accessed 21-Sep-2023].
- I Calculated ChatGPT’s IQ. <https://www.youtube.com/watch?v=HXb9Azzhr1k>. Accessed: 2023-10-30.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. 2023. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*.
- Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, et al. 2023. Empowering practical root cause analysis by large language models for cloud incidents. *arXiv preprint arXiv:2305.15778*.
- Haakon Faste and Honray Lin. 2012. The untapped promise of digital mind maps. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1017–1026.
- Xidong Feng, Ziyu Wan, Muning Wen, 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*.
- Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Christian Petersen, Alexis Chevalier, and Julius Berner. 2023. Mathematical capabilities of chatgpt. *arXiv preprint arXiv:2301.13867*.
- Aurélien Garivier and Eric Moulines. 2011. On upper-confidence bound policies for switching bandit problems. In *International Conference on Algorithmic Learning Theory*, pages 174–188. Springer.
- 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*.
- Peter Jamieson. 2012. Using modern graph analysis techniques on mind maps to help quantify learning. In *2012 Frontiers in Education Conference Proceedings*, pages 1–6. IEEE.
- Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. 2023. Causal reasoning and large language models: Opening a new frontier for causality. *arXiv preprint arXiv:2305.00050*.

- Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. 2023. [Reward design with language models](#). In [The Eleventh International Conference on Learning Representations](#).
- Yuxi Li. 2017. Deep reinforcement learning: An overview. [arXiv preprint arXiv:1701.07274](#).
- Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. 2023. Making ppo even better: Value-guided monte-carlo tree search decoding. [arXiv preprint arXiv:2309.15028](#).
- Jieyi Long. 2023. Large language model guided tree-of-thought. [arXiv preprint arXiv:2305.08291](#).
- Xuefei Ning, Zinan Lin, Zixuan Zhou, Huazhong Yang, and Yu Wang. 2023. Skeleton-of-thought: Large language models can do parallel decoding. [arXiv preprint arXiv:2307.15337](#).
- Reham Omar, Omij Mangukiya, Panos Kalnis, and Esam Mansour. 2023. Chatgpt versus traditional question answering for knowledge graphs: Current status and future directions towards knowledge graph chatbots. [arXiv preprint arXiv:2302.06466](#).
- OpenAI. 2023. [Gpt-4 technical report](#).
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. [Advances in Neural Information Processing Systems](#), 35:27730–27744.
- Martin L Puterman. 1990. Markov decision processes. [Handbooks in operations research and management science](#), 2:331–434.
- Christopher D Rosin. 2011. Multi-armed bandits with episode context. [Annals of Mathematics and Artificial Intelligence](#), 61(3):203–230.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. [arXiv preprint arXiv:1707.06347](#).
- Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. [arXiv preprint arXiv:2303.11366](#).
- David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. [nature](#), 550(7676):354–359.
- Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. 2023. Gpt-4 doesn't know it's wrong: An analysis of iterative prompting for reasoning problems. [arXiv preprint arXiv:2310.12397](#).
- Yashar Talebirad and Amirhossein Nadiri. 2023. Multi-agent collaboration: Harnessing the power of intelligent llm agents. [arXiv preprint arXiv:2306.03314](#).
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. [arXiv preprint arXiv:2307.09288](#).
- Miles Turpin, Julian Michael, Ethan Perez, and Samuel R Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. [arXiv preprint arXiv:2305.04388](#).
- Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In [Chi conference on human factors in computing systems extended abstracts](#), pages 1–7.
- Karthik Valmeekam, Matthew Marquez, and Subbarao Kambhampati. 2023. Can large language models really improve by self-critiquing their own plans? [arXiv preprint arXiv:2310.08118](#).
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In [The Eleventh International Conference on Learning Representations](#).
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. [Advances in Neural Information Processing Systems](#), 35:24824–24837.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. [arXiv preprint arXiv:2305.10601](#).
- Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. [arXiv preprint arXiv:2210.03493](#).
- Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. 2023. Multi-modal chain-of-thought reasoning in language models. [arXiv preprint arXiv:2302.00923](#).

## A Task Setup

An overview of these tasks is provided in Table 9.

### A.1 Game of 24

We collect a dataset from (4nu), comprising 1,362 games ranked by human solving time, spanning a range of difficulty levels from easy to hard. For our testing phase, we randomly selected 137 games, ensuring coverage of various difficulty intervals. The remaining 1,225 problems were used to train the policy/value networks with MCTS. In the context of this task, as outlined in Table 1, the thoughts refer to the three intermediate equations, while the state encompasses the available numbers (ranging from 1 to 4) for creating the equations. Actions involve the selection of two numbers and an operator to form an equation, and the reward is set to 1 if the final equation is both valid and results in the number 24, utilizing each of the input numbers exactly once, otherwise it is set to -1. Performance is measured by calculating the success rate across the 137 test games.

### A.2 8-Puzzle

We randomly generated 419 solvable 8-puzzle problems, with 300 instances allocated for training and 119 instances for testing. All generated problems are solvable within 9 steps. The action space encompasses four directions: [Up, Down, Left, Right]. Note that the legal action space for each problem state may vary due to the dynamic position of the empty space. As shown in Table 1, the thoughts refer to the step-by-step move, and the puzzle state after the move.

### A.3 Pocket Cube

We initially set all faces of the cube to a uniform color and then randomly apply 5 actions sequentially selected from the 27 legal actions of the Rubik’s Cube. This process resulted in the creation of 1,000 training samples and 183 testing samples. All generated problems can be solved within 4 steps. To simplify the action space, we reduced the 27 legal operations to 9 actions, namely: {U, U’, U2, R, R’, R2, F, F’, F2}, which are used in our experiments with both baselines and XoT. As shown in Table 1, the thoughts pertain to the step-by-step rotation, and the cube state after the move.

## B Baselines & XoT Setup

**GPT-3.5 / GPT-4.** We note that temperature and top\_p are set to 0.0 for all LLM invoked to minimize inference variance.

**LLaMA-2-13B (finetuned).** To evaluate the potential of directly distilling knowledge from simulations into a smaller model to possibly avoid using a large model like GPT-4 during testing, we fine-tuned the LLaMA-2-13B model. Our experiments were carried out on eight V100 GPUs, each with 80GB of memory, and lasted approximately 5 hours. The training setup involved 5 epochs, a train batch size of 32, an evaluation batch size of 1, and a single step for gradient accumulation. The evaluation and save strategies were set to "no" and "steps" respectively, with saving occurring every 20 steps and a limit of one saved model. The learning rate was  $2e-5$ , with no warmup steps and logging every 2 steps. We employed a cosine learning rate scheduler. By using ground truth labels—considered more accurate than labels from MCTS simulations—we aimed to convert an optimization or search problem into a more straightforward prediction or supervised learning challenge, using a training dataset of <question, answer> pairs.



### B.1 Game of 24

The IO prompt is supported by five in-context examples. In the case of CoT, we augment each input-output pair by including three intermediate equations. As for ToT, we solicit one-step thought candidates from the LLM at each step, subsequently instructing the LLM to categorize each thought candidate for intermediate selection. For experimental comparison, we conduct experiments on both the top-1 candidate (with  $b=1$ ) and the top-3 candidates (with  $b=3$ ) being retained, where  $b$  indicates the branches retained for exploration at each step. For GoT, we employ LLM to generate one-step thought candidates in the same manner as ToT, then we direct the LLM to select the top-1 thought from all candidates for merging the thoughts. We also examine a CoT-SC baseline, which derives the majority output from 10 CoT samples. For XoT, we perform 200 simulations for each action taken, and this count is increased to 500 during the thought revision process.

In the multi-solution scenario, the IO, CoT, and CoT-SC prompts each include 5 examples, with each problem having 1 to 3 different solutions. For ToT, the top-3 candidates (with  $b=3$ ) at the final



Table 9: An overview of tasks employed in this study.

	<b>Game of 24</b>	<b>8-Puzzle</b>	<b>Pocket Cube</b>
<b>Objective</b>	Use four numbers on playing cards to make the number 24 through +, −, ×, or ÷.	Rearrange the tiles in the 3 × 3 puzzle from an scrambled state to a goal state $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ .	Rotating the faces of a 2 × 2 pocket cube until each face of the cube is a uniform color  .
<b>Input</b>	4 numbers ranging from 1 to 13, <i>e.g.</i> , (4, 6, 10, 10).	A scrambled 3 × 3 digital puzzle, <i>e.g.</i> , $\begin{bmatrix} - & 4 & 7 \\ 5 & 3 & 8 \\ 6 & 2 & 1 \end{bmatrix}$ .	A scrambled 2 × 2 pocket cube, <i>e.g.</i> ,  . Colors represented as numbers for LLMs.
<b>Output</b>	An equation to reach 24, <i>e.g.</i> , $4 \times 6 + 10 - 10 = 24$ .	The slide sequence of the “-” tile, <i>e.g.</i> , (Up, Down, Left, Right ···).	The rotation move sequence of the cube, <i>e.g.</i> , (F, R2, U’ ···).
<b>Thought</b>	3 intermediate equations.	The step-by-step sliding, and the puzzle state after the move.	The step-by-step rotation, and the cube state after the move.
<b>State</b>	The remaining 1-4 numbers.	The current number layout of the puzzle.	Colors of each face of the pocket cube.
<b>Action</b>	Picking two number and a operation to compose an equation.	The one-step moving action of the “-” tile.	The one-step rotation action of cube.
<b>Reward</b>	1 if the number of the final number is equal to 24 otherwise -1.	The negative minimum step on solving the current puzzle state toward the goal state.	The negative minimum moving step on solving current cube state toward the goal state.

step are considered as different solutions. Rather than keeping only the top-1 thought, GoT is instructed to select between 1 to 3 thoughts from all candidates at each step to generate a wider range of solutions. As for XOT, after performing simulations on MCTS, we sample 500 thought trajectories as for exploration and remove duplicates. The top-3 thoughts with the highest counts are preserved.

## B.2 8-Puzzle

The IO prompt is extended with three in-context examples. In the CoT approach, each input-output pair is enriched by incorporating intermediate legal action sets, the current action, and the current state. In ToT, at each stage, a set of one-step thought candidates are derived from the LLM, from the current set of legal actions. We impose a maximum step limit of 9 since all generated problems can be solved within this range. The 8-puzzle’s rules are conveyed through a system message, including detailed explanations of each action’s execution. Similarly, we perform 20 simulations for each action taken with XOT, and increase this number to 50 for thought revision processes.

In the multi-solution scenario, all of the IO, CoT,

and CoT-SC prompts consist of four examples. Each problem is presented with one to three distinct solutions. For ToT (b=3) and GoT (k=3), the maximum number of steps is increased to 12, as correct solutions may not always be optimal and could exceed 9 steps. In the case of XOT, after conducting simulations with MCTS, we sample 50 thought trajectories for exploration and select the top-3 thoughts with the highest counts.

## B.3 Pocket Cube

The IO prompt is augmented with a single in-context example. In CoT, we enrich each input-output pair by including intermediate actions and states. In ToT, we retrieve one-step thought candidates from the LLM at each stage and instruct the LLM to classify each candidate for intermediate selection. A maximum step limit of 4 is imposed, as all generated problems can be resolved within this range. The cube’s rules are conveyed through a system message, which includes the definition of the action space and illustrations of the execution of each action. For XOT, we conduct 20 simulations for each action taken and increase it to 500 for revision.

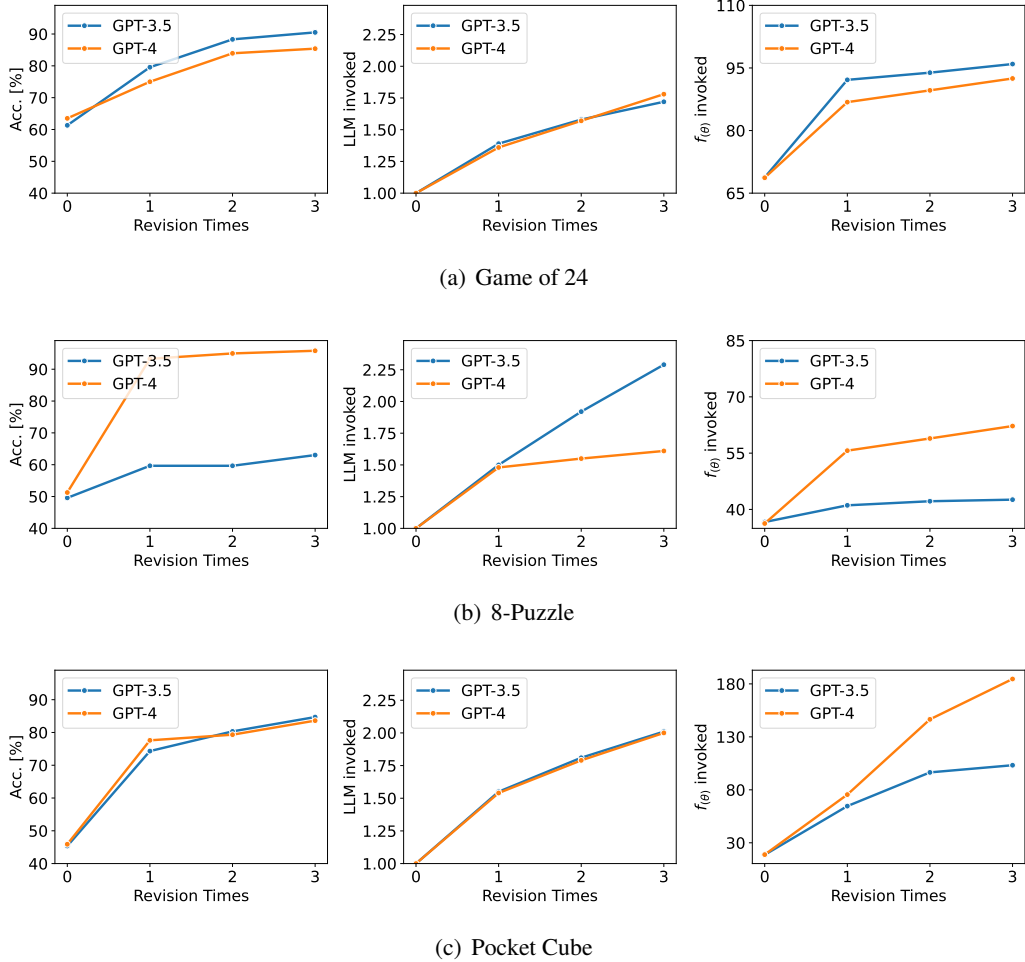


Figure 5: Accuracy, LLM and  $f_{\theta}$  invoked comparison on XOT w.r.t. the number of revisions.

In the multi-solution setup, the IO, CoT, and CoT-SC prompts each include 3 examples, and each problem within these prompts offers 3 unique solutions. As for ToT ( $b=3$ ) and GoT ( $k=3$ ), the maximum number of steps allowed is extended to 7. In the case of XOT, after conducting MCTS simulations, we gather 50 thought trajectories, and we keep the top 3 thoughts with the highest counts.

### C Policy / Value Networks Configurations

The policy and value networks in our model utilize a shared multi-layer perceptron (MLP) architecture with two layers and hidden units arranged as (128, 256). Two heads connected to the MLP are responsible for predicting  $v_{\theta}(s)$  and  $P_{\theta}(s)$  separately. The total number of parameters in the Policy/Value Network for all three tasks is approximately  $10^6$ . This design results in a considerably smaller model compared to LLM, making it much more efficient. We train this model through three iterations, with each iteration comprising 10 self-play episodes for

MCTS.

## D Ablation Study

In our ablation study, we consider two aspects: the impact of the number of revisions on the performance and efficiency of XOT and the sensitivity of performance to the completeness of the provided thoughts. These angles allow us to gain insights into how XOT’s performance can be improved and understand the importance of providing complete thoughts in complex problem-solving tasks.

### D.1 Number of Revisions

It’s important to highlight that the performance of each task can be further improved through multiple revisions of the thought using the MCTS-LLM collaborative framework. In Fig. 5, we compare the performance of GPT-3.5 and GPT-4 models using the XOT method with varying numbers of revisions, ranging from 0 to 3, across all three tasks.

In the Game of 24 task, as the number of re-

Table 10: Performance comparison on three tasks with incomplete thoughts.

Task	Model	GPT-3.5			GPT-4		
		Acc. [%]	LLM invoked	$f_\theta$ invoked	Acc. [%]	LLM invoked	$f_\theta$ invoked
Game of 24	ToT (b=1)	3.65	17.15	-	40.88	18.55	-
	GoT (k=1)	2.19	5.00	-	9.49	5.00	-
	<b>XoT (w/o revise)</b>	<b>17.52</b>	<b>1.00</b>	<b>68.73</b>	<b>43.07</b>	<b>1.00</b>	<b>68.70</b>
8-Puzzle	ToT (b=1)	0.00	32.60	-	6.72	26.98	-
	GoT (k=1)	0.00	18.63	-	3.36	19.00	-
	<b>XoT (w/o revise)</b>	<b>2.52</b>	<b>1.00</b>	<b>36.66</b>	<b>40.34</b>	<b>1.00</b>	<b>36.24</b>
Pocket Cube	ToT (b=1)	0.55	16.48	-	2.19	16.39	-
	GoT (k=1)	0.00	8.96	-	1.64	8.68	-
	<b>XoT (w/o revise)</b>	<b>5.46</b>	<b>1.00</b>	<b>18.85</b>	<b>6.01</b>	<b>1.00</b>	<b>18.89</b>

visions increases, both models exhibit improved performance. Notably, GPT-3.5 consistently outperforms GPT-4 in terms of accuracy. After three revisions, GPT-3.5 achieves an accuracy of 90.51%, while GPT-4 reaches 85.40%. This improved performance comes at the cost of increased inference times and model calls, primarily driven by the need for more interactions to generate revised thoughts. For the 8-Puzzle task, the trend of increasing accuracy with more revisions remains valid. However, in this task, GPT-4 significantly outperforms GPT-3.5. After one revision, GPT-4 achieves an accuracy of 93.28%, which increases to 95.80% after the third revision. In contrast, GPT-3.5 only attains an accuracy of 63.03% after the third revision. In the Pocket Cube task, the performance trend is similar. The accuracy of both models improves with an increase in the number of revisions. GPT-3.5 starts at an accuracy of 45.36% without revision and improves to 84.70% after three revisions. GPT-4 begins with an accuracy of 45.90% and reaches 83.61% after three revisions. Inference times and model calls are comparable between the two models, with GPT-4 showing a substantial increase in model calls after the third revision.

Note that the number of LLM invocations does not increase dramatically with additional revisions, even though  $f_\theta$  is called more times to guide simulations. Considering the significant disparity in inference costs between LLM and  $f_\theta$ , increasing the number of revisions to achieve better performance appears to be a favorable trade-off.

Table 11: Revision Success Rate for GPT-3.5.

Revisions	Game of 24	8-Puzzle	Pocket Cube
XoT (w/ 1 r)	47.17%	20.00%	53.00%
XoT (w/ 2 r)	69.81%	21.31%	63.64%
XoT (w/ 3 r)	75.93%	26.67%	72.00%

We also focus on the efficacy of the revision process within the XoT framework across three

Table 12: Revision Success Rate for GPT-4.

Revisions	Game of 24	8-Puzzle	Pocket Cube
XoT (w/ 1 r)	32.69%	85.96%	58.59%
XoT (w/ 2 r)	55.10%	89.47%	60.00%
XoT (w/ 3 r)	60.00%	91.38%	70.00%

distinct tasks. The Revision Success Rate is calculated as the ratio of successfully detected errors to the number of failed cases without revision, thereby providing insight into the effectiveness of revisions. The results for both GPT-3.5 and GPT-4 are presented in Table 11 and Table 12.

Our observations reveal a high revision success rate in the XoT framework, which increases with the number of revisions. This underscores the effectiveness of LLMs in the revision process, positioning it as a highly efficient approach to thoughts revision.

## D.2 Incomplete Thought

In this ablation study, we explore the performance of LLMs when provided with incomplete thoughts, specifically omitting the last step of the thought trajectory. This simulates scenarios where MCTS might supply inaccurate or incomplete thoughts. The aim is to test whether LLMs can independently solve problems or rely on their own reasoning, rather than solely relying on the thought from MCTS as answers. We present the performance comparison for all three tasks in Table 10. Note that we only compare ToT and GoT since other baselines do not support this comparison by their nature.

The results clearly show that incomplete thoughts lead to a significant performance drop in all three tasks. GPT-3.5 is more affected than GPT-4, with GPT-3.5 achieving 0% accuracy on several baselines. In contrast, XoT with GPT-4 attains satisfactory performance on the Game of 24 and 8-Puzzle, achieving over 40% accuracy. However,

the performance of XOT is dramatically affected in the Pocket Cube task, with accuracy dropping to 6%. This demonstrates that for very complex tasks, LLMs are highly sensitive to the completeness of the thoughts provided. Missing steps in the thought can lead to a substantial drop in performance, highlighting the importance of providing complete thoughts for such tasks.

## E Use of AI Assistants

We acknowledge the utilization of ChatGPT for refining the paper and assisting in the coding process. However, it is imperative to clarify that none of the content within the paper was generated by ChatGPT.

## F Prompt Example

Prompts 1-3 display example CoT prompts utilized for Game of 24, 8-Puzzle, and Pocket Cube. These templates are applicable to CoT, ToT, GoT, and our XOT in the final inference process. Each thought step includes the action taken and the resulting new state.

### Instruction: Game of 24

Use numbers and basic arithmetic operations (+ - \* /) to obtain 24.

### Prompt: Game of 24

**Input:** 2 9 10 12

**Steps:**

12 \* 2 = 24 (left: 9 10 24) Expression: 9, 10, (12) \* (2)

10 - 9 = 1 (left: 24 1) Expression: (12) \* (2), (10) - (9)

1 \* 24 = 24 (left: 24) Expression: ((10) - (9)) \* ((12) \* (2))

Answer: (12 \* 2) \* (10 - 9) = 24

### Revision: Game of 24

Using the given [input] numbers and basic arithmetic operations (+, -, \*, /), follow the steps strictly to achieve a result of 24.

All the [input] numbers can reach 24 by basic arithmetic operations (+, -, \*, /).

If the final answer is not exactly 24, then the corresponding [Steps] is considered [wrong]. Please help me identify the exact

wrong step based on its left number, among [Step 1, Step 2, Step 3]. If you are uncertain about which step is wrong, please begin your analysis with [Step 1] for better understanding.

**Input:** 2 9 10 12

**Steps:**

[Steps 1] 12 \* 2 = 24 (left: 9 10 24) Expression: 9, 10, (12) \* (2)

[Steps 2] 24 - 10 = 14 (left: 9 14) Expression: 9, ((12) \* (2)) - (10)

[Steps 3] 9 + 14 = 23 (left: 23) Expression: (9) + ((12) \* (2)) - (10)

The Steps are wrong. Because it can not reach 24 in the end. To be specific, 23 is not equal to 24. [Steps 2] is wrong. Because it is impossible to reach 24 from the step 2. After Step 2, left numbers are 9, 13.

$$9 + 13 = 22$$

$$9 * 13 = 111$$

$$9 - 13 = -4$$

It is impossible to reach 24 from [Steps 2].

### Instruction: 8-Puzzle

You are a virtual expert in solving a 8-puzzle problem. Please follow the instructions and rules below to complete the solving. Your goal is to reach the goal state with valid moves.

[The goal state]

0 1 2

3 4 5

6 7 8

[Instructions]

The 8-puzzle consists of a 3x3 grid containing 8 numbered tiles (from 1 to 8) and one empty space (denoted by 0). Only 0 can be moved horizontally or vertically, and the objective is to reach the goal state from a given initial state. The goal state is typically the numbers ordered sequentially, with the 0 in the first position:

[The goal state]

0 1 2

3 4 5

6 7 8

[Rules]



1. Only 0 can be moved horizontally or vertically.

2. Each move is chosen from the following set of options:

- 'Left': move 0 to the left
- 'Down': move 0 downward
- 'Right': move 0 to the right
- 'Up': move 0 upward

For example:

Before move:

1 2 3

4 0 6

7 8 5

After move 'Left':

1 2 3

0 4 6

7 8 5

Before move:

1 2 3

4 0 6

7 8 5

After move 'Down':

1 2 3

4 8 6

7 0 5

Before move:

1 2 3

4 0 6

7 8 5

After move 'Right':

1 2 3

4 6 0

7 8 5

Before move:

1 2 3

4 0 6

7 8 5

After move 'Up':

1 0 3

4 2 6

7 8 5

3. The next move must be chosen from the valid move set depending on the position of '0'.

For example:

p1 p2 p3

p4 p5 p6

p7 p8 p9

(1) If '0' is located at position 'p1', the valid move set is ['Right', 'Down'].

(2) If '0' is located at position 'p2', the valid move set is ['Left', 'Right', 'Down'].

(3) If '0' is located at position 'p3', the valid move set is ['Left', 'Down'].

(4) If '0' is located at position 'p4', the valid move set is ['Right', 'Up', 'Down'].

(5) If '0' is located at position 'p5', the valid move set is ['Left', 'Right', 'Up', 'Down'].

(6) If '0' is located at position 'p6', the valid move set is ['Left', 'Up', 'Down'].

(7) If '0' is located at position 'p7', the valid move set is ['Right', 'Up'].

(8) If '0' is located at position 'p8', the valid move set is ['Left', 'Right', 'Up'].

(9) If '0' is located at position 'p9', the valid move set is ['Left', 'Up'].

4. Diagonal moves are not allowed.

5. The objective is to return the moves which can reach the goal state.

#### Prompt: 8-Puzzle

All given problems can be solved within 1 to 9 steps. The next move must be chosen from the valid move set. The maximum step number you can take is 9. Try to reach the goal state using the least number of steps ( $\leq 9$ ). **\*\*DO NOT exceed 9 steps.\*\***

#### [Initial State]:

3 1 2

6 4 5

7 8 0

#### [Process]:

3 1 2

6 4 5

7 8 0

**Step 1:** Choose one valid move from: [Left, Up]

Move: Left

Current State:

3 1 2

6 4 5

7 0 8

**Step 2:** Choose one valid move from: [Left, Right, Up]

Move: Left

Current State:

3 1 2

6 4 5

0 7 8

**Step 3:** Choose one valid move from:

[Right, Up]

Move: Up

Current State:

3 1 2

0 4 5

6 7 8

**Step 4:** Choose one valid move from:

[Right, Up]

Move: Up

Current State:

0 1 2

3 4 5

6 7 8

**Finished.**

**[Moves]:**

Left, Left, Up, Up

6 4 5

0 7 8

**Step 3:** Choose one valid move from:

[Right, Up]

Up

3 1 2

0 4 5

6 7 8

**Step 4:** Choose one valid move from:

[Right, Up]

Right

3 1 2

4 0 5

6 7 8

Finished.

The given [Process] is not correct because number 3, 4, 0, 5 are not their goal positions in the end. The puzzle has failed on reaching its goal state.

Now please help me identify the exact step number that is wrong. You must provide one wrong step. If you can not provide an exact step number, please consider that it could be "all steps are wrong".

[Step 4] is wrong, with Move: Right.

#### Revision: 8-Puzzle

The given [Process] is not correct since it does not reach the goal state in the end.

If the final answer does not reach the goal state, then the corresponding [Process] is considered [wrong]. Please help me identify the exact wrong step based on its left number, among [Step 1, Step 2, Step 3, ...]. If you are uncertain about which step is wrong, please begin your analysis with [Step 1] for better understanding.

Please help me identify the exact step number that is wrong. You must provide one wrong step.

**[Initial State]:**

3 1 2

6 4 5

7 8 0

**[Process]**

3 1 2

6 4 5

7 8 0

**Step 1:** Choose one valid move from: [Left,

Up]

Left

3 1 2

6 4 5

7 0 8

**Step 2:** Choose one valid move from: [Left,

Right, Up]

Left

3 1 2

#### Instruction: Pocket Cube

You are a virtual expert in solving a 2x2 Pocket Cube. Your task is to restore a scrambled 2x2 Rubik's Cube to its original state. All the given problems can be solved in 1 to 4 moves. You cannot exceed more than 11 moves. Provide the sequence of moves required for the restoration. Please follow the instructions and rules below to complete the solving:

1. A 2x2 Pocket Cube has six faces, namely: [Upper, Front, Bottom, Left, Right, Back] Each consisting of a 2x2 grid of squares, with each square having its own color.

2. Colors in the Cube are represented in numbers: [0, 1, 2, 3, 4, 5]

3. The Cube's state is represented into a facelets expanding graph, for instance:

Upper:

0 0

0 0

Front:

5 5

2 2

Down:

3 3

3 3

Left:

1 1

4 4

Right:

4 4

1 1

Back:

2 2

5 5

4. A restoration of a Pocket Cube is to move squares in each face to have same numbers.

Some example Restored States are:

[Restored State]

Upper:

0 0

0 0

Front:

2 2

2 2

Down:

3 3

3 3

Left:

4 4

4 4

Right:

1 1

1 1

Back:

5 5

5 5

Or

[Restored State]

Upper:

2 2

2 2

Front:

0 0

0 0

Down:

3 3

3 3

Left:

1 1

1 1

Right:

4 4

4 4

Back:

5 5

5 5

You must make move to the Cube to achieve a Restored State, not limited to the above one. Note that we just need each face to have same numbers, no matter which face has which color.

5. You are only allowed to use following moves [U, U', U2, R, R', R2, F, F', F2].

["U": Turn the Upper face of the cube 90 degrees clockwise. For instance, after taking move U:

Upper:

0 0

0 0

Front:

2 2

2 2

Down:

3 3

3 3

Left:

4 4

4 4

Right:

1 1

1 1

Back:

5 5

5 5

will become

Up:

0 0

0 0

Front:

1 1

2 2

Down:

3 3

3 3

Left:

2 2 4 4

Right:

5 5

1 1

Back:

4 4 5 5

"U": Turn the Upper face of the cube 90 degrees counterclockwise (or anti-clockwise).

For instance, after taking move U':

Upper:

0 0

0 0

Front:

2 2

2 2

Down:

3 3

3 3

Left:

4 4

4 4

Right:

1 1

1 1

Back:

5 5

5 5

will become

Upper:

0 0

0 0

Front:

4 4

2 2

Down:

3 3

3 3

Left:

5 5

4 4

Right:

2 2

1 1

Back:

1 1

5 5

"U2": Turn the Upper face of the cube 180 degrees (a half turn). For instance, after taking move U2:

Upper:

0 0

0 0

Front:

2 2

2 2

Down:

3 3

3 3

Left:

4 4

4 4

Right:

1 1

1 1

Back:

5 5

5 5

will become

Up:

0 0

0 0

Front:

5 5

2 2

Down:

3 3

3 3

Left:

1 1

4 4

Right:

4 4

1 1

Back:

2 2

5 5

"R": Turn the Right face of the cube 90 degrees clockwise. For instance, after taking move R:

Upper:

0 0

0 0

Front:

2 2

2 2

Down:

3 3

3 3

Left:

4 4

4 4

Right:

1 1

1 1

Back:

5 5



5 5  
 will become  
 Upper:  
 0 2  
 0 2  
 Front:  
 2 3  
 2 3  
 Down:  
 3 5  
 3 5  
 Left:  
 4 4  
 4 4  
 Right:  
 1 1  
 1 1  
 Back:  
 0 5  
 0 5  
 "R'": Turn the Right face of the cube 90 de-  
 grees counterclockwise. For instance, after  
 taking move R':  
 Upper:  
 0 0  
 0 0  
 Front:  
 2 2  
 2 2  
 Down:  
 3 3  
 3 3  
 Left:  
 4 4  
 4 4  
 Right:  
 1 1  
 1 1  
 Back:  
 5 5  
 5 5  
 will become  
 Upper:  
 0 5  
 0 5  
 Front:  
 2 0  
 2 0  
 Down:  
 3 2

3 2  
 Left:  
 4 4  
 4 4  
 Right:  
 1 1  
 1 1  
 Back:  
 3 5  
 3 5  
 "R2": Turn the Right face of the cube 180  
 degrees. For instance, after taking move R':  
 Upper:  
 0 0  
 0 0  
 Front:  
 2 2  
 2 2  
 Down:  
 3 3  
 3 3  
 Left:  
 4 4  
 4 4  
 Right:  
 1 1  
 1 1  
 Back:  
 5 5  
 5 5  
 will become  
 Up:  
 0 3  
 0 3  
 Front:  
 2 5  
 2 5  
 Down:  
 3 0  
 3 0  
 Left:  
 4 4  
 4 4  
 Right:  
 1 1  
 1 1  
 Back:  
 2 5  
 2 5

"F": Turn the Front face of the cube 90 degrees clockwise. For instance, after taking move F:

Upper:  
0 0  
0 0  
Front:  
2 2  
2 2  
Down:  
3 3  
3 3  
Left:  
4 4  
4 4  
Right:  
1 1  
1 1  
Back:  
5 5  
5 5  
will become

Up:  
0 0  
4 4  
Front:  
2 2  
2 2  
Down:  
1 1  
3 3  
Left:  
4 3  
4 3  
Right:  
0 1  
0 1  
Back:  
5 5  
5 5

"F'": Turn the Front face of the cube 90 degrees counterclockwise. For instance, after taking move F':

Upper:  
0 0  
0 0  
Front:  
2 2  
2 2  
Down:  
3 3

3 3  
Left:  
4 4  
4 4  
Right:  
1 1  
1 1  
Back:  
5 5  
5 5  
will become

Upper:  
0 0  
1 1  
Front:  
2 2  
2 2  
Down:  
4 4  
3 3  
Left:  
4 0  
4 0  
Right:  
3 1  
3 1  
Back:  
5 5  
5 5

"F2": Turn the Front face of the cube 180 degrees. For instance, after taking move F2:

Upper:  
0 0  
0 0  
Front:  
2 2  
2 2  
Down:  
3 3  
3 3  
Left:  
4 4  
4 4  
Right:  
1 1  
1 1  
Back:  
5 5  
5 5  
will become

Upper:  
0 0  
3 3  
Front:  
2 2  
2 2  
Down:  
0 0  
3 3  
Left:  
4 1  
4 1  
Right:  
4 1  
4 1  
Back:  
5 5  
5 5

Prompt: Pocket Cube

All the given problems can be solved in 1 to 4 moves. \*\*You cannot exceed more than 11 moves.\*\* Please complete [Process] and return the [Restoration Moves].

**[Initial Cube State]:**

Upper:  
4 5  
4 4  
Front:  
5 1  
5 0  
Down:  
0 0  
2 0  
Left:  
1 1  
3 2  
Right:  
2 2  
4 3  
Back:  
3 3  
1 5

**[Process]:**

**[Step 1]**

[Move] R  
[Current Cube State]  
Upper:  
4 0  
4 0

Front:  
5 5  
0 1  
Down:  
0 1  
2 2  
Left:  
1 1  
3 3  
Right:  
2 2  
4 3  
Back:  
4 3  
5 5  
**[Step 2]**  
[Move] U'  
[Current Cube State]

Upper:  
0 0  
4 4  
Front:  
0 1  
0 1  
Down:  
2 2  
2 2  
Left:  
1 1  
3 3  
Right:  
4 3  
4 3  
Back:  
5 5  
5 5

**[Step 3]**

[Move] F'  
[Current Cube State]  
Upper:  
0 0  
0 0  
Front:  
1 1  
1 1  
Down:  
2 2  
2 2  
Left:  
3 3

3 3

Right:

4 4

4 4

Back:

5 5

5 5

**Finished.**

Now strictly follow the above process to form Restoration Moves.

**[Restoration Moves]:**

R U' F'

### Revision: Pocket Cube

The given [Process] is not correct since it does not reach the goal state in the end.

If the final answer does not reach the goal state, then the corresponding [Process] is considered [wrong]. Please help me identify the exact wrong step based on its left number, among [Step 1, Step 2, Step 3, ...]. If you are uncertain about which step is wrong, please begin your analysis with [Step 1] for better understanding.

Please help me identify the exact step number that is wrong. You must provide one wrong step.

**[Initial Cube State]:**

Upper:

4 5

4 4

Front:

5 1

5 0

Down:

0 0

2 0

Left:

1 1

3 2

Right:

2 2

4 3

Back:

3 3

1 5

**[Process]:**

**[Step 1]**

[Move] R

[Current Cube State]

Upper:

4 0

4 0

Front:

5 5

0 1

Down:

0 1

2 2

Left:

1 1

3 3

Right:

2 2

4 3

Back:

4 3

5 5

**[Step 2]**

[Move] U'

[Current Cube State]

Upper:

0 0

4 4

Front:

0 1

0 1

Down:

2 2

2 2

Left:

1 1

3 3

Right:

4 3

4 3

Back:

5 5

5 5

**[Step 3]**

[Move] F2

[Current Cube State]

Upper:

0 0

1 1

Front:

2 2

2 2

Down:

4 4



3 3

Left:

4 0

4 0

Right:

3 1

3 1

Back:

5 5

5 5

Finished.

After finishing all the moves: The Upper face still has 2 different colors. The Down face still has 2 different colors. The Left face still has 2 different colors. The Right face still has 2 different colors.

The given [Process] is not correct because not every face has the same numbers in the end. The cube has failed on restoring to its original state. Now please help me identify the exact step number that is wrong. You must provide one wrong step. If you can not provide an exact step number, please consider that it could be "all steps are wrong". [Step 3] is wrong, with Move: F2.