

On the Empirical Complexity of Reasoning and Planning in LLMs

Anonymous ACL submission

Abstract

Chain-of-thought (CoT), tree-of-thought (ToT), and related techniques work surprisingly well in practice for some complex reasoning tasks with Large Language Models (LLMs), but why? This work seeks the underlying reasons by conducting experimental case studies and linking the performance benefits to well-established sample and computational complexity principles in machine learning. We experimented with 6 reasoning tasks, ranging from grade school math, air travel planning, . . . , to Blocksworld. The results suggest that (i) both CoT and ToT benefit significantly from task decomposition, which breaks a complex reasoning task into a sequence of steps with low sample and computational complexity and explicitly outlines the reasoning structure, and (ii) for computationally hard reasoning tasks, the more sophisticated tree structure of ToT outperforms the linear structure of CoT. These findings provide useful guidelines for the use of LLM in solving reasoning tasks in practice.

1 Introduction

Reasoning and planning tasks are often challenging due to their inherently multi-step processes. Compared with the **Direct** approach that prompts the model to provide immediate answers, large language models (LLMs) showed surprising results on reasoning problems when they were asked to explain their reasoning step-by-step through a **chain-of-thought (CoT)** (Wei et al., 2022; Kojima et al., 2022) before providing their answers. This was followed by improvements of the **tree-of-thought (ToT)** (Yao et al., 2023; Xie et al., 2023), combining LLMs with a search algorithm to structure reasoning steps into a tree and selecting promising next steps by self-evaluation. Other variants include **CoT with self-consistency (CoT-SC)** (Wang et al., 2022), which generates multiple CoTs and responds with the most common answer.

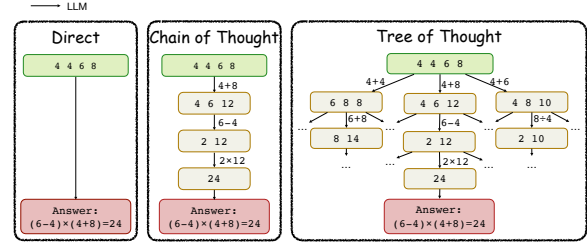


Figure 1: An illustration of LLM reasoning methods on the Game of 24. Given four poker cards, the player combines the cards using basic arithmetic operations, (+, -, ×, ÷), to reach the target number of 24.

However, evidence shows that the relative performance of CoT, ToT, and their variants may vary from task to task. For example, CoT and its variants, such as self-consistency (CoT-SC) (Wang et al., 2022), have been successful in solving grade school math problems (GSM). Our experiments show that ToT is only marginally better than CoT and even performs worse than CoT-SC on GSM when using the same token budget. However, in the Game of 24 shown in Fig 1, where four numbers need to be manipulated with arithmetic operations to obtain the number 24, using CoT to provide a solution with a short reasoning chain fails badly. In addition, CoT-SC did not show significant improvement while using more tokens at inference. In contrast, ToT significantly outperforms CoT and CoT-SC on Game of 24 (Yao et al., 2023). These pieces of evidence raise an important question: what are the underlying factors that affect the effectiveness of different LLM reasoning methods in various tasks?

To answer this question, we investigate when and why CoT and ToT are effective in reasoning and planning problems from the viewpoint of sample complexity and computational complexity in machine learning. Sample complexity captures the amount of data needed for learning predictors, which we measure by description length (Shai and Shai, 2014), the number of bits required to describe the learnable part of the predictor. The predictors take an input, e.g., a sequence of words, and produce a prediction, e.g. a label that may be used directly or

as a component of a larger reasoning process (Wei et al., 2022). Predictors with a small description length can be shown to require less training data, i.e. a small sample complexity (Shai and Shai, 2014), to achieve low generalisation error. Computational complexity is relevant in two distinct ways in this paper: 1) in the amount of computation required for learning, e.g. finding the correct parameters in the predictor given the training data, and 2) in the amount of computation required for reasoning, e.g. finding the solution given a problem. Learning may become computationally intractable if the values of hidden variables are not observed during learning¹ (Aloise et al., 2009; Blum and Rivest, 1988), motivating us to consider their presence during learning of CoT. For problems that are computationally hard to solve, e.g. NP-hard problems, it is unlikely that a small predictor producing a short CoT solution exists in the worst case, which motivates the use of more complex structures like search trees.

We focus on the **reasoning** and **planning** problems in the context of **natural language processing**. A reasoning problem entails deducing the answer to a question from provided evidence and applicable reasoning rules. It often requires applying various rules multiple times to connect different pieces of evidence and draw a conclusion. Planning, a special subset of reasoning, requires an action sequence to achieve a desired goal state from a current state. Planning often requires reasoning over a long time horizon, making it computationally harder to solve.

We empirically study these issues through six case studies: **grade school mathematics** (Cobbe et al., 2021), **multi-hop question answering** (Trivedi et al., 2022), a simple **dynamic programming (DP)** problem (Dziri et al., 2023), **air travel planning** (Zhao et al., 2023), **Game of 24** (Yao et al., 2023) and **Blocksworld** (Valmeekam et al., 2023)². Grade school math, multi-hop question answering and DP problems we consider have computationally efficient reasoning components. Air travel planning has two different efficient solutions that we compare. Finally, Game of 24 and Blocksworld appears to be computationally difficult.

We study the problems under different settings, including prompting general LLMs, fine-tuning with task-specific data, and in-context learning with task-specific examples. Our main findings are consistent over different settings. These findings

¹Indirectly the values need to be inferred during learning.

²Due to lack of space, we describe the multi-hop question answering and Blocksworld case studies in Appendix B.

suggest a few guiding principles for using LLM to solve reasoning and planning tasks in practice:

- *CoT and ToT enhances LLM reasoning when problem decomposition lowers the sample complexity.* In all six cases, decomposing problems with chain or tree structures reduces sample complexity and improves performance.
- *Explicitly annotating necessary information improves CoT performance.* In the DP problem and Blocksworld, explicitly demonstrating relevant variables improves learning. For tasks with small search trees like air travel planning, CoT that linearizes the search tree and explicitly describing intermediate computations outperforms CoT that directly predicts the next action.
- *Tree structures help when generating short-chain solutions is computationally hard; CoT-SC is effective otherwise.* For tasks like Game of 24 and Blocksworld, where short-chain solutions are likely computationally hard to find, ToT works better. In tasks like GSM and multi-hop QA with efficient algorithms for finding next steps, CoT-SC performs better.

2 Related Work

LLMs have shown significant progress in tackling reasoning and planning problems. Initial studies (Wei et al., 2022; Wang et al., 2022; Kojima et al., 2022; Chen et al., 2022; Gao et al., 2023) unveiled various prompting techniques that enable LLMs to demonstrate reasoning processes step by step, thereby substantially boosting their reasoning abilities. This approach has been swiftly adapted to address everyday planning issues (Huang et al., 2022a,b; Ahn et al., 2022; Song et al., 2023; Wang et al., 2023; Singh et al., 2023). Subsequent research has integrated LLMs with diverse search algorithms, further enhancing their capability to solve complex reasoning and planning challenges (Zhang et al., 2023; Yao et al., 2023; Zhao et al., 2023; Xie et al., 2023; Ding et al., 2023; Feng et al., 2023; Hao et al., 2023; Liu et al., 2023).

Several works explore why CoT improves performance. Feng et al. (2024) investigates how CoT enhances model capacity; Prystawski et al. (2024) and Wang et al. (2024) examine the training data distribution that enables CoT to excel; Dziri et al. (2023) discusses CoT’s limitations in compositional reasoning. Our work takes a different approach, we explore principles that guide LLM behavior across various reasoning frameworks, and provide insights into selecting appropriate strategies for different

tasks. While Zhao et al. (2023) discusses the sample complexity of LLM planning, they overlook the computational implications.

3 Analysis of LLM Reasoning Methods

We aim to analyze LLM reasoning methods from the sample and computational complexity perspective in machine learning. Our goal is to understand the complexity of problems using simple representations. However, analyzing LLM learning with transformer architecture is challenging, and the effects of pre-training, which we do not control, are present throughout. Instead, we empirically observe whether our analysis reflects the practical behaviour of LLMs and whether the insights from our analysis are practically useful. Specifically, when our analysis suggests that a particular method is preferred, we examine whether it is indeed preferred empirically.

3.1 Problem formulation

The LLM reasoning approach, e.g., CoT and ToT, essentially uses a sequential decision approach to do the reasoning instead of other approaches, such as converting to a Boolean satisfiability problem (SAT) and solving it using SAT solvers. Thus, we formulate reasoning and planning problems using planning terminology. A planning problem can be defined using a state space S , an action space A , a transition function T , and a goal function G . The state space S defines all possible states (e.g., the boolean values of assigned variables for logical inference or scalar values of assigned variables for math problems). The action space A consists of possible equations or rules. Each state has applicable actions that can be executed. The transition function $T(s, a)$ specifies a new state s' after applying action a in state s . The goal function G specifies whether the current state is a goal state.

The solution to a reasoning or planning problem is a sequence of actions in the action space that transforms the initial state into a goal state. In a CoT, we directly use the LLM as a **policy** to map the current state (as inferred by the LLM from the context) to the action, while in a ToT, the LLM is used to specify applicable actions in each state to construct a search tree. LLM is also used as a transition function in both methods.

3.2 Decomposition and sample complexity

3.2.1 Description length (DL)

Description length (DL), the number of bits required to describe the learnable part of the predictor, is

used to analyse the sample complexity in machine learning. It is used in the minimum description length (MDL) principle, also known as Occam’s Razor from the philosophy of science. The MDL principle suggests that a method with a shorter description length requires less training data and is preferred. MDL has been formalized in various ways. One formal statement (from section 7.3 of Shai and Shai (2014)) is provided here:

Theorem 3.1 (Occam’s Razor) *Let \mathcal{H} be a hypothesis class and let $d: \mathcal{H} \rightarrow \{0, 1\}^*$ be a prefix-free description language for \mathcal{H} . Then, for every sample size, m , every confidence parameter, $\delta > 0$, and every probability distribution, D , with probability greater than $1 - \delta$ over the choice of $S \sim D^m$ we have that, $\forall h \in \mathcal{H}, L_D(h) \leq L_S(h) + \sqrt{(|h| + \ln(2/\delta))/2m}$, where $L_S(h)$ is the empirical loss of h on the S , $L_D(h)$ is the expected loss of h , and $|h|$ is the length of $d(h)$.*

According to Theorem 3.1, we can bound the expected loss of a solution h by the description length $|h|$ and the training loss $L_S(h)$. Thus, when trained by the same amount of data, predictors with smaller DL have lower generalisation errors, indicating a lower sample complexity.

3.2.2 DL analysis of LLM reasoning methods

For each reasoning method, we can analyze the description length of its policy and transition.

Direct Direct answering method does not involve explicit steps of reasoning, all reasoning is performed internally in the neural network. This method may have a low sample complexity if the neural network architecture closely aligns with the reasoning algorithm (Xu et al., 2020). Analysing whether the algorithm fits the inductive bias of the network is complex, so we mostly explore a tabular representation for simplicity. In problems with N variables, each taking K values, direct answers require learning a table of size K^N , which exponentially increases with more variables.

CoT With $|A|$ number of possible actions, each depending on a_i variables, the description length for transition functions of these actions is proportional to $\sum_{i=1}^{|A|} K^{a_i}$. We also need a *policy* function predicting action to select based on observations with its description length of K^M if it depends on M variables. If the policy depends only on whether the variables have been observed rather than their values, then a binary table of size 2^M is sufficient. CoT-SC runs CoT multiple times and responds with the most common answer – this reduces the

prediction variance and, consequently, the sample complexity at the cost of more computation.

ToT The complexity of transition functions in ToT is analysed similarly to CoT. Another component is the self-evaluation module, which evaluates whether the current state or selected action is promising to reach the final answer. It acts as a search heuristic: the highly promising branches will more likely be selected for the next search steps. This self-evaluation module is essentially a classifier. Similar to the policy for next-step prediction, the classifier can be described by a table with a size of K^M if it depends on M variables to determine the next actions. If the policy depends only on whether the variables have been observed rather than their values, then a binary table of size 2^M is sufficient.

Our analysis is mostly done assuming a tabular representation. This is further complicated by two factors: the amount of pre-training of the LLMs, which we do not control, and possible failures in training, e.g. due to the presence of local minimums. Empirical observations in our case studies assess whether the transformer architecture used in LLM resembles table-filling behaviour for each case or successfully learns a better algorithm, taking into account the effects of pre-training.

3.3 Reasoning structure and computational complexity

The computational complexity of solving (versus learning) a reasoning or planning problem is a key factor in choosing between CoT and ToT. Some problems, e.g. NP-complete problems, have short solutions that are verifiable in polynomial time, but efficient policies to find such solutions are unlikely to exist. ToT, with its search algorithm, presents a viable approach for such hard problems by allowing the use of more computation during the search process. However, for simple problems with low computational complexity, a complex search algorithm may not produce additional improvement.

4 Case Studies³

4.1 Grade School Maths

GSM8K (Cobbe et al., 2021) consists of grade school math problems in natural language. LLMs solve these problems effectively with CoT (Achiam et al., 2023). The dataset is widely used to evaluate the reasoning ability of language models. We analyze the dataset and find that it is well-suited for CoT, as most problems can be solved with a linear-

³See Appendix A and I for experimental details and complete prompts.

time policy. We also empirically show that CoT is usually sufficient, and ToT offers little improvement. In a subset of 50 randomly selected problems, we found that 49 can be solved with a chain-style algorithm where each step involves selecting an equation with all but one variable known, allowing the remaining variable’s value to be inferred. The remaining problem, solvable with simultaneous equations, is ignored in the rest of the study.

4.1.1 Analysis

Direct Consider a problem with N variables each can take K values. A tabular representation requires a table of size K^N and description length of $O(K^N \log K)$ for each question type, assuming answers also take K possible values (thus $\log K$ bits to represent). The description length can be large (we manually analyze a subset of GSM8K problems and show statistics in Appendix C.1).

CoT In a math word problem, the action can be selecting the next applicable rule, and the transition would be deriving the next value of an intermediate variable based on the rule and the value of the input variables. Assuming A different actions whose transition functions require a_i variables, the total description length of the learnable transitions would be $O(\sum_{i=1}^A K^{a_i} \log K)$. This is more manageable than Direct (see Appendix C.1 for details). As for the policy, we can select an equation where the values of all except one variable are known. There exists a linear time forward chaining algorithm to do that (Appendix C.3), which translates to a relatively small policy that needs to be learned. The components of the decomposed problems are relatively simple and suggest that decomposition with CoT may be reasonable for this problem.

ToT As discussed in CoT, there is a simple policy for deciding the next equation to solve. If the policy is learned reasonably well by the LLM, a search may give limited improvement.

4.1.2 Experiments

In-context Learning Our experimental results are shown in Fig. 2a. We see that GPT-3.5 and GPT-4 give lower performance than CoT due to the higher sample complexity, which aligns with our analysis.

An unanalysed factor is that LLMs still need to learn to extract the equations from the question, learn the world knowledge that is not included in the question, and ground the variables’ values from the previous observations. The LLMs, particularly GPT-4, do remarkably well on GSM8K, indicating that extraction and grounding may not be major

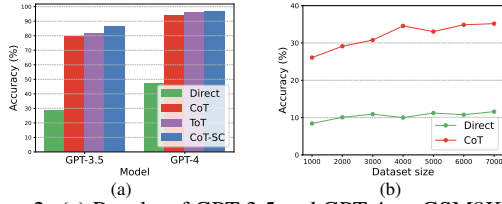


Figure 2: (a) Results of GPT-3.5 and GPT-4 on GSM8K Test set; (b) Fine-tuning results on Llama2-7b

hurdles for LLMs. Some errors are still present and are discussed in the Appendix C.4.

For ToT, we run a beam search ToT, branching after each sentence on the choice of the next sentences suggested by the LLM. We prompt the same LLM to self-evaluate the quality of each proposed reasoning step. ToT yields similar performance to CoT, indicating the policy is reasonably well-learned. We also see that using the same token budget, CoT-SC performs better than ToT since it reduces the variance of the predictions. This suggests that when the problem does not inherently require search, spending the inference budget on CoT-SC may be more beneficial than ToT.

Fine-tuning The GPT experiments suggest that LLMs have difficulties learning to solve GSM-type questions directly. To check that, we perform fine-tuning experiments using the GSM8K training set and compare models fine-tuned with direct answer completion and CoT completion.

We fine-tune with varying sizes of subsets of the GSM8K training set and test performance on the GSM8K test set. The results are shown in Fig. 2b. Note that each CoT example provides substantially more information than each Direct example, but CoT is substantially better even when Direct is provided with 7 times more training examples (Direct at 7k vs CoT at 1k). It suggests that the transformer in the LLM is behaving more like a tabular predictor and cannot learn to decompose the problem internally without being trained explicitly to do so.

4.2 Dynamic Programming

We study another problem, the Maximum Weighted Independent Set problem (MWIS) (Kleinberg and Tardos, 2005): *Given a sequence of integers, find a subsequence with maximum sum such that no two elements in the subsequence are adjacent in the original sequence.* The problem can be solved in linear time using dynamic programming (see Appendix D.1). MWIS was studied in Dziri et al. (2023), showing that LLMs trained on short sequences generalize poorly to longer sequences. In this paper, we focus on the amount of annotation provided in learning where only the answer is provided

in Direct, whereas different levels of explicitness in annotation can be provided in CoT.

4.2.1 Analysis

Direct Consider a sequence with N integers; each may take K values. A tabular representation would have K^N entries, where each entry needs N bits to indicate the presence of the N number in the subsequence, giving a description length of $O(NK^N)$. **CoT** Using concepts in planning, we can see CoT as a combination of selecting which function to calculate next (policy), and derive the intermediate results based on the function and input variables (transition). There are a constant number of unique function that may take up to 3 variables (see Appendix I.2 for examples), so the description length of the transition would be $O(K^3 \log K)$, which represents a mapping from input variables to the output value ($\log K$ bits to represent). To decide which function to apply next, we only need to know what variables have been calculated and what have not, so the policy can be represented by a table of size $O(N)$. Overall, the description length of CoT would be $O(K^3 \log K + N)$, which appears manageable.

4.2.2 Experiments

In-context Learning In this section, we will compare prompting LLMs to answer the MWIS problem directly with prompting them to answer using CoT. We will also study two versions of CoT demonstrations and show that a more explicit demonstration can improve performance substantially.

Consider the following line from the CoT demonstration (see I.3 for the entire demonstration):

Implicit prompt (from Dziri et al. (2023)): ... *Since* $dp[0] \neq input[0] + dp[2]$ ($6 \neq -4 + 5$) ...

We can make it more explicit as follows:

Explicit prompt: ... *Since* $dp[0]=6$, $input[0]=-4$, $dp[2]=5$, $input[0] + dp[2] = 1 \neq 6 = dp[0]$...

Both prompts demonstrate steps to use DP to solve the MWIS problem, but in the Implicit prompt, when autoregressively generating the token "!=", the values of $dp[0]$, $input[0]$, $dp[2]$, and $input[0]+dp[2]$ are not explicitly stated in the immediate context and need to be inferred from all previous observations.

As shown in Fig. 3, making the demonstrations explicit provides more than 20% improvement compared to the implicit demonstrations from Dziri et al. (2023). This is consistent with the learning problem becoming computationally easier if all relevant variables are explicit during learning. The sample complexity may also be smaller, as explicit demon-

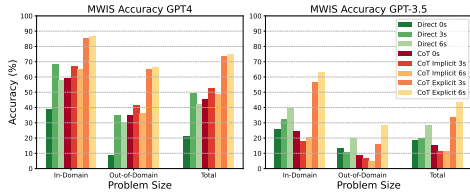


Figure 3: In-context learning results on MWIS. 3-shot prompts have one example each for sizes 4, 5, and 6, while 6-shot prompts have two examples each. "In-domain" refers sizes 4, 5, and 6, and "Out-of-Domain" refers to sizes from 6 to 10.

strations decompose the single reasoning steps into multiple simpler steps, effectively creating a small CoT. In contrast, we can view deciding between " \neq " and " $=$ " in the implicit demonstration as a function of all the previously observed variables. The tabular representation of such a function has a large description length which suggests that it would require a larger sample complexity to learn.

We observe that prompting LLM to directly give an answer performs comparably to the implicit CoT method (Fig. 3). This suggests that while we prompt the LLM to "directly" give an answer, the underlying transformer model is not necessarily learning it by populating a table of size K^N as it is unlikely to encounter a very large number of examples of the MWIS problem during pre-training. This suggests that the transformer used in the LLM may align well with the reasoning algorithm used here. We explore this further in fine-tuning experiments.

Fine-tuning We perform fine-tuning experiments to study both in- and out-of-domain performance.

To examine the generalizability of the fine-tuned model to OOD examples, we define two types of Domain: 1) *Problem size*: Fine-tune with problems of sizes 4, 5, and 6. Test with problems of size ranging from 4 to 10. All numbers in the input array are uniformly sampled from $[-100, 100]$ 2) *Number range*: Fine-tune and test with problems of sizes 4, 5, and 6. For fine-tuning data, numbers in the input array are uniformly sampled from $[-100, 100]$, while for OOD test examples, numbers are uniformly sampled from $[-1000, 1000]$.

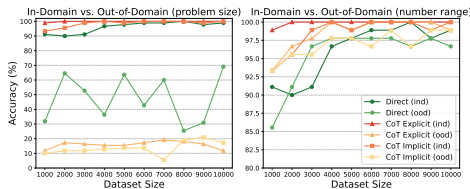


Figure 4: Results of fine-tuning Llama2-7B-chat on MWIS.

Results For in-domain test examples, CoT Explicit outperforms CoT Implicit and Direct with the same number of training examples. Interestingly, with more fine-tuning data, Direct can achieve performance similar to CoT Explicit. This contrasts with

math word problems, where Direct is not comparable to CoT even with 10 times more fine-tuning data. Training the transformer to directly approximate a DP algorithm seems easier than to compute the result of a multivariate equation in math word problems. However, it is unclear if this difficulty is due to computational complexity in learning or poor alignment of the transformer with solving the equation; see Appendix E for more discussion.

As shown in Dziri et al. (2023), CoT struggles to generalize to reasoning lengths longer than the training data, performing worse than Direct, likely due to LLMs relying on pattern matching rather than compositional learning (Dziri et al., 2023; Kharitonov and Chaabouni, 2020). However, all methods generalize well to different ranges of numbers, possibly because the solution structure remains the same, making pattern matching less of an issue.

4.3 Air Travel Planning

Consider the problem of planning an air trip: *given the starting city and destination, provide the flight route using direct flights between cities.* For example, *What is the flight route from Singapore to New Orleans?* A valid answer might be *Singapore-San Francisco-Houston-New Orleans.* This is a typical graph search problem where nodes are cities and edges are direct flights, and we aim to find a valid path that connects the start and target cities. To solve it, we can either use an LLM to predict the route directly or leverage the LLM's knowledge of the flight graph for a graph search. Zhao et al. (2023) studied this using prompting. In this paper, we go further and linearize the graph search algorithm into a CoT to study fine-tuning and learning of the graph search algorithm.

4.3.1 Analysis

Assume there are N cities. We randomly select two cities as the current and target cities. We repeat the description length analysis in Zhao et al. (2023), then extend it to a linearized ToT.

Direct & CoT Generating the path directly is essentially the same as CoT, as we generate the cities on the path autoregressively. A concise representation of it is a table, where rows and columns are the current and goal cities, and each entry records the next city to fly to reach the goal. This table has N^2 entries, each taking $\log N$ bits to describe, resulting in a description length of $O(N^2 \log N)$ bits.

ToT In ToT, LLM acts as the graph, i.e., predicts the direct flight from the current city, together with a hand-coded breadth-first search (BFS) algorithm to find the valid route. Assuming that the total num-

ber of edges grows proportionally to the number of cities, describing a sparse graph with N nodes takes approximately $O(N \log N)$ bits, with $\log N$ bits to describe each city in the adjacency list. The graph describes the transition functions; thus, ToT can be described using $O(N \log N)$ bits if the other components are hand-coded. We can linearize the BFS algorithm into a CoT which is entirely generated by the LLM. Other than providing the adjacent cities to each city, the components include being maintaining a first-in-first-out queue, checking whether a city has been visited and recognizing the goal city. For a sparse graph as described, the runtime of BFS is $O(N)$, which translates to the existence of relatively small predictors for all the functions.

4.3.2 Experiments

Since Direct and CoT are essentially the same, we compare CoT with ToT experimentally. For ToT, the LLM is used only in the expansion step of BFS, when it is queried to generate the neighbour of a city. In addition, we linearize the ToT process into a CoT by generating all the intermediate steps in the BFS computation in **ToT-linear**.

We evaluate the settings of travelling between large cities (with a population of more than 5 million) and mid-sized cities (with a population between 1 million and 5 million). Details of the dataset can be found in Appendix F.

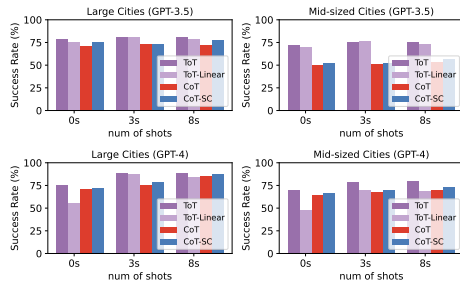


Figure 5: Results of GPT-3.5 and GPT-4 in air travel planning.

In-context learning The result for in-context learning is shown in Fig. 5. For GPT-3.5, ToT outperforms CoT slightly in large cities and substantially in mid-sized cities. This is consistent with the analysis where the description length of CoT and ToT are $O(N^2 \log N)$ and $O(N \log N)$ respectively: the gap between CoT and ToT would be larger when N is larger. Surprisingly, ToT-linear is comparable to ToT, even for zero-shot, where the steps in the BFS algorithm are briefly described in the prompt without any examples of its execution, indicating that there is some pre-training of the BFS algorithm in GPT-3.5. GPT-4 generally does better than GPT-3.5 for ToT and CoT, possibly because it has been trained with more data. Interestingly, GPT-4

does not do so well for ToT-linear, particularly for zero-shot, indicating that its pre-training for the BFS algorithm is possibly poorer than GPT-3.5.

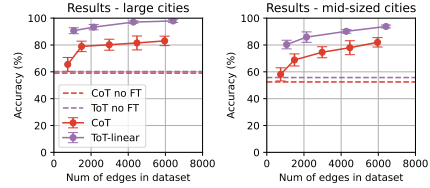


Figure 6: Results of fine-tuning Llama2-7b using different dataset sizes. The *CoT no FT* and *ToT no FT* means using the pre-trained Llama2-7b with CoT and ToT.

Fine-tuning We perform fine-tuning experiments which allows us to better control the amount of training data used in the experiments. The results of our fine-tuning experiments are in Fig. 6. Each ToT-linear example is longer than a CoT example; hence, we plot the results based on the number of edges observed in training. The results are consistent with the complexity analysis, with ToT-linear performing better than CoT. ToT-linear can also be viewed as a CoT where the intermediate computations are explicitly described instead of being left for the LLM to learn implicitly.

4.4 Game of 24

Many puzzles like Game of 24 are designed to be hard and unlikely to be efficiently solvable (Kendall et al., 2008), although we are not aware of results on the computational complexity of the Game of 24⁴. We use the Game of 24 shown in the introduction: given four numbers, the player must use basic arithmetic operations (+, −, ×, ÷) and all four numbers to reach 24. These types of puzzle games are often designed to be hard to solve.

The results in Yao et al. (2023), obtained with in-context learning, show that CoT fails while ToT does substantially better. We extend the results by showing that CoT fails in fine-tuning as well, suggesting that the failure is likely due to the mismatch between the computational structure of CoT and the problem. We also consider the decomposition of the actions for in-context learning and show that the decomposition of complex actions into a sequence of simpler actions within a ToT can lead to substantial improvement in performance.

4.4.1 Analysis

We provide a general form of Game of 24 for analysis. Assume N numbers are given, and each

⁴A modified version with N rather than four numbers, arbitrary target number instead of 24, and only addition and multiplication with zero allowed is the same as subset-sum, an NP-complete problem. This suggests that similar puzzles are computationally difficult to solve.

number can take K different values. The goal is to use those numbers with arithmetic operations (+, −, ×, ÷) to reach T . For the standard Game of 24, $N = 4$, $T = 24$.

Direct Represented as a table, there are K^N inputs. A solution is an expression consisting of the N numbers together with $N - 1$ operations and corresponding parentheses. Assuming $\log K$ bits to represent numbers, this can be represented using $O(N \log K)$ bits, giving a total table size of $O(K^N N \log K)$ bits.

CoT For CoT, the $N - 1$ operations are produced in a step-by-step manner. Each step has $N(N - 1)/2$ ways to select two numbers and 6 distinct operations (two orderings for − and ÷), giving $3N(N - 1)$ possible actions. The transition of each operation can be represented with a table with K^2 entries using $O(K^2 \log K)$ bits, although pre-training likely has learned these operations for small K . This gives a total description length of $O(N^2 K^2 \log K)$ if each action is learned using its own table. If we decompose the selection of two numbers and the arithmetic operation into two steps, then the total description length is $O(N^2 + K^2 \log K)$, and we consider this decomposition in our experiments (see Appendix I.5.1 for examples). Like other computationally difficult problems, there is no simple known policy for selecting the next action. A simple tabular policy would have $O(K^N)$ entries, each described using $O(\log N)$ bits.

ToT ToT uses the same actions as CoT but does not need a policy. Instead, we have a goal recognizer and an evaluation function that decides which nodes to expand. Verifying whether a solution is correct can be done in $O(N)$ time, hence a goal recognizer with a small representation exists. Difficult computational problems typically do not have a simple evaluation function for intermediate steps; a tabular evaluation function would have $O(K^N)$ entries. However, a ToT may use a larger computation budget to search a larger part of the search tree when the evaluation function is weaker, compared to CoT, where the next action is selected with a fixed learned policy.

4.4.2 Experiments

We perform fine-tuning with 1200 solution trajectories of Game of 24 to demonstrate the difficulty of learning a small chain solution. Both CoT and Direct failed in all test cases, indicating that moderate amounts of data are insufficient for learning in these settings. For in-context learning, the success rate of the 100 games is reported in Fig. 7.

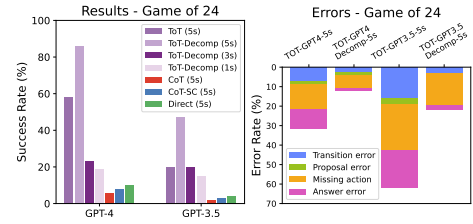


Figure 7: Results of Game of 24. 5s, 3s and 1s means 5, 3, and 1 examples in the prompt for few-shot in-context learning.

For ToT, we use a beam search with a beam width of 5 and the same action and self-evaluation prompts as Yao et al. (2023). We also constructed a more decomposed version of ToT, ToT-Decomp, where we decompose the action into two steps: the selection of two numbers and the arithmetic operation. Also, ToT-Decomp uses a small CoT that provides the steps for constructing the final equation from the sequence of actions and states in the solution, whereas ToT directly generates the final equation from the action-state sequence.

From the results we see that ToT clearly outperforming CoT and Direct. Additionally, ToT-Decomp significantly outperforms ToT, demonstrating the advantages of decomposition even within the components of ToT. Error analysis (Fig. 7) categorizes the errors into four types: 1) transition error – the next state (remaining numbers) is generated incorrectly; 2) proposal error – the LLM does not generate the correct numbers in the action expression; 3) missing actions – there are valid actions but are not proposed by the LLM; and 4) answer error – the search is correct but the final expression is incorrect. ToT-Decomp notably reduces each error type compared to ToT.

5 Conclusion

This paper provides a detailed empirical study to understand the effectiveness of CoT and ToT reasoning in planning and reasoning tasks using notions of sample and computational complexity in machine learning. We view CoT and ToT as decomposition methods for the underlying problem and study the complexity of the component predictors in the decomposed problems. Our study finds that when the solution can be decomposed as a chain of reasoning steps where predicting the next step is not difficult, explicitly demonstrating the reasoning chain during learning can be helpful. Leaving out important variables for deciding the next reasoning step instead of making all relevant variables explicit in the demonstrations will make learning more difficult. Finally, when algorithmic analysis indicates that predicting the next reasoning step in a CoT is computationally hard, a ToT structure can be helpful.

Limitations The suggested methodology from this paper is to analyse the chain-of-thought as a decomposition of the problem and to analyse the complexity of its components. If learning the components has low sample complexity and the computational complexity of predicting the next reasoning step is low, then learning to solve the problem using a chain-of-thought would be reasonable. On the other hand, if the computational complexity of predicting the next reasoning step is high, it may be reasonable to consider learning the components and using a tree-of-thought to solve the problem. This oversimplifies various aspects of the problem. Even though the components have low sample complexity, it may be difficult to learn them in practice as the computational complexity of learning may be high, although this may be alleviated by overparameterization of the predictors used to learn the components. Another issue is out-of-domain generalization. As shown in the MWIS case study, generalization in-domain does not mean that the method will generalize out-of-domain, which may be further exacerbated by overparameterization. Further limitations may apply when doing in-context learning where very few examples are used. Performance may depend heavily on the pre-trained LLM used in this setting. Nonetheless, our case studies suggest that the proposed methodology may still be useful in the in-context learning setting. We would suggest using the guidelines proposed in this paper in a similar way that the Occam Razor principle in the philosophy of science is used. Occam’s Razor suggests that simple explanations for a scientific phenomenon be preferred until shown otherwise by observations. The suggestions we proposed may not work all the time but should similarly be preferred until empirical observations suggest otherwise.

Ethics Statement This paper studies reasoning and planning in LLMs from a general perspective. While we do not focus on ethics issues, reasoning and planning techniques can potentially be useful in ensuring that AI agents behave ethically through the use of appropriate reward or goal functions that may possibly be learned from data. They may also be used in harmful ways in planning more sophisticated attacks against others. Research on both the use of reasoning and planning for ensuring ethical AI agent behaviour and in mitigating the use of reasoning and planning in performing harmful attacks should be encouraged.

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*. 787
- Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*. 788
- Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. 2009. Np-hardness of euclidean sum-of-squares clustering. *Machine learning*, 75:245–248. 789
- Avrim Blum and Ronald Rivest. 1988. Training a 3-node neural network is np-complete. *Advances in neural information processing systems*, 1. 790
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*. 791
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*. 792
- Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. Everything of thoughts: Defying the law of penrose triangle for thought generation. *arXiv preprint arXiv:2311.04254*. 793
- William F Dowling and Jean H Gallier. 1984. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming*, 1(3):267–284. 794
- Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D Hwang, et al. 2023. Faith and fate: Limits of transformers on compositionality. *arXiv preprint arXiv:2305.18654*. 795
- Guhaog Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2024. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36. 796
- 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*. 797
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham 798

840	Neubig. 2023. Pal: Program-aided language models. In <i>International Conference on Machine Learning</i> , pages 10764–10799. PMLR.	894
841		895
842		896
843	Naresh Gupta and Dana S Nau. 1992. On the complexity of blocks-world planning. <i>Artificial intelligence</i> , 56(2-3):223–254.	897
844		898
845		899
846	Shibo Hao, Yi Gu, Haotian Luo, Tianyang Liu, Xiyan Shao, Xinyuan Wang, Shuhua Xie, Haodi Ma, Adithya Samavedhi, Qiyue Gao, et al. 2024. Llm reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models. <i>arXiv preprint arXiv:2404.05221</i> .	900
847		901
848		902
849		903
850		904
851		905
852	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. <i>arXiv preprint arXiv:2305.14992</i> .	906
853		907
854		908
855		909
856	Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. <i>arXiv preprint arXiv:2106.09685</i> .	910
857		911
858		912
859		913
860	Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022a. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In <i>International Conference on Machine Learning</i> , pages 9118–9147. PMLR.	914
861		915
862		916
863		917
864		918
865	Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. 2022b. Inner monologue: Embodied reasoning through planning with language models. <i>arXiv preprint arXiv:2207.05608</i> .	919
866		920
867		921
868		922
869		923
870		924
871	Graham Kendall, Andrew Parkes, and Kristian Spoerer. 2008. A survey of np-complete puzzles. <i>ICGA Journal</i> , 31(1):13–34.	925
872		926
873		927
874	Eugene Kharitonov and Rahma Chaabouni. 2020. What they do when in doubt: a study of inductive biases in seq2seq learners. In <i>International Conference on Learning Representations</i> .	928
875		929
876		930
877		931
878	Jon Kleinberg and Eva Tardos. 2005. <i>Algorithm Design</i> . Addison-Wesley Longman Publishing Co., Inc., USA.	932
879		933
880	Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. <i>Advances in neural information processing systems</i> , 35:22199–22213.	934
881		935
882		936
883		937
884		938
885	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. <i>arXiv preprint arXiv:2309.15028</i> .	939
886		940
887		941
888		942
889		943
890	Ben Prystawski, Michael Li, and Noah Goodman. 2024. Why think step by step? reasoning emerges from the locality of experience. <i>Advances in Neural Information Processing Systems</i> , 36.	944
891		945
892		946
893		947
		948
	Shalev-Shwartz Shai and Ben-David Shai. 2014. <i>Understanding machine learning: From theory to algorithms</i> . Cambridge university press.	
	Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2023. Prog-prompt: Generating situated robot task plans using large language models. In <i>2023 IEEE International Conference on Robotics and Automation (ICRA)</i> , pages 11523–11530. IEEE.	
	Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In <i>Proceedings of the IEEE/CVF International Conference on Computer Vision</i> , pages 2998–3009.	
	Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. <i>arXiv preprint arXiv:2307.09288</i> .	
	Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. $\hat{\text{a}}^{\text{TM}}\ll \text{MuSiQue}$: Multi-hop questions via single-hop question composition. <i>Transactions of the Association for Computational Linguistics</i> , 10:539–554.	
	Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the planning abilities of large language models—a critical investigation. <i>Advances in Neural Information Processing Systems</i> , 36:75993–76005.	
	Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. <i>arXiv preprint arXiv:2305.16291</i> .	
	Xinyi Wang, Alfonso Amayuelas, Kexun Zhang, Liangming Pan, Wenhua Chen, and William Yang Wang. 2024. Understanding the reasoning ability of language models from the perspective of reasoning paths aggregation. <i>arXiv preprint arXiv:2402.03268</i> .	
	Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. <i>arXiv preprint arXiv:2203.11171</i> .	
	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. <i>Advances in Neural Information Processing Systems</i> , 35:24824–24837.	
	Gail Weiss, Yoav Goldberg, and Eran Yahav. 2021. Thinking like transformers. In <i>International Conference on Machine Learning</i> , pages 11080–11090. PMLR.	

949 Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao,
950 Min-Yen Kan, Junxian He, and Qizhe Xie. 2023.
951 Self-evaluation guided beam search for reasoning.
952 In *Thirty-seventh Conference on Neural Information*
953 *Processing Systems*.

954 Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-
955 ichi Kawarabayashi, and Stefanie Jegelka. 2020. What
956 can neural networks reason about? In *International*
957 *Conference on Learning Representations*.

958 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran,
959 Thomas L Griffiths, Yuan Cao, and Karthik
960 Narasimhan. 2023. Tree of thoughts: Deliberate
961 problem solving with large language models. *arXiv*
962 *preprint arXiv:2305.10601*.

963 Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu
964 Ding, Joshua B Tenenbaum, and Chuang Gan. 2023.
965 Planning with large language models for code gener-
966 ation. *arXiv preprint arXiv:2303.05510*.

967 Zirui Zhao, Wee Sun Lee, and David Hsu. 2023.
968 Large language models as commonsense knowl-
969 edge for large-scale task planning. *arXiv preprint*
970 *arXiv:2305.14078*.

971 Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin,
972 Omid Saremi, Joshua M. Susskind, Samy Bengio,
973 and Preetum Nakkiran. 2024. [Understanding length](#)
974 [generalization by thinking like transformers](#). In *The*
975 *Twelfth International Conference on Learning Repre-*
976 *sentations*.

977
978
979
980
981
982
983
984
985
986
987
988
989
990
991

992
993
994
995
996
997
998
999
1000
1001

1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025

A Experimental Details

All prompting experiments are done with gpt-3.5-turbo-1106 and gpt-4-1106-preview. All fine-tuning experiments are done with Llama2-7B-chat (Touvron et al., 2023) with LoRA $r = 64$, $\alpha = 16$ (Hu et al., 2021) applied to query and value matrices, and uses $batch_size = 1$ and gradient accumulation steps= 32. MWIS and Game of 24 are fine-tuned for 5 epochs with a learning rate of $3e - 4$. Travel planning is fine-tuned for 300 gradient optimization steps with a learning rate of $3e - 4$. The fine-tuning data is wrapped in the template "`<s> [INST] {{prompt}} [/INST] {{completion}} </s>`" and the loss is calculated on completion tokens.

B Additional case studies

B.1 Multi-hop Question Answering

We study the MusiQue dataset (Trivedi et al., 2022), where the task is to identify the answer given a question and a context of up to 20 paragraphs. The question may look like "*Who did the hitman from The Hitman’s Bodyguard play in Star Wars?*", which can be solved by decomposing the question and answering each decomposed question to get the answer.

B.1.1 Analysis

This question answering task requires both natural language processing and reasoning. We describe the reasoning process as follows: the paragraphs in the context provide a set of relation triplets, and the question can be translated to a logical expression which is a conjunction of some relation triplets, with several unknown entities in it, one of which is the target answer. The reasoning process would require finding the unknown entities in some order and to derive the final answer. The natural language processing part is common to different reasoning methods, while the complexity of reasoning can be different.

Direct The set of relation triplets represented by the context can form a knowledge graph, with head and tail entities in the triplets being vertices and relations being edges. Consider a problem where there are V vertices and E edges in the context knowledge graph and question, each vertex and edge may take K different values. To answer the question directly would require memorizing a table of size $K^{(V+E)} \log K$, representing a mapping from possible configurations of the question and

graph to an answer. Thus the description length of learning Direct in a tabular manner would be $O(K^{(V+E)} \log K)$.

CoT Similar to the Grade School Maths problem, the reasoning process in this task can be seen as a sequence of actions and transitions. The action here would be to select a triplet in the question with unknown entities, and the transition is to infer the unknown entity. The unknown entity can be found by matching the known entity and relation from the question triplet to the appropriate triplet in the context. The same entity can be named differently in the question and the context triplets. A table of size K^2 can be learned to specify whether two entities match. Matching relations is similar. This suggests that learning the transition is manageable with reasonable complexity, although extracting the entity using natural language sentences adds additional complexity. A policy for this problem can simply be selecting a triplet in the question that has two known element and one unknown element. As in the Grade School Maths problem, this can be done via forward chaining in time linear in the size of the knowledge graph. The small policy representation suggests that learning the policy would not have high sample complexity.

ToT The description length of the transition functions of ToT is the same as CoT and as in the CoT case, performance of the transition likely depends more on the natural language processing component. In addition, all single hop questions in the dataset have one unique answer. Hence, there is no real need for branching if the transition is well learned, and search may offer little improvement.

B.1.2 Experiments

We are not able to fit each question which is paired with 20 paragraphs into the context window of Llama-2 for fine-tuning. Instead, we perform in-context learning studies with GPT models. We use 6-shot demonstrations consisting of the question and answer (which includes the reasoning steps for CoT) but did not include the contexts of the 6 questions in the demonstrations as the length would be too long. Results are shown in Figure 8. The experimental results align with our analysis. We make the following observations: 1) CoT consistently outperforms Direct but the gap between them is smaller when using GPT4. One possible explanation is that, the policy for reasoning, i.e. selecting the next triplet to infer its unknown entity, is relatively simple, and the more powerful GPT-4

1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059

1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076

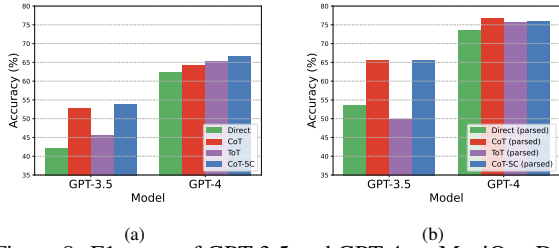


Figure 8: F1 score of GPT-3.5 and GPT-4 on MusiQue Dev set. (a) is using natural language context, (b) is using LLM parsed relation triplets as context.

may have been able to learn this directly during training. Extracting the correct entities from the context becomes a common bottleneck for the two methods. 2) ToT performs similarly to CoT when using GPT-4. This aligns with our analysis that ToT may give little improvement as there is no branching in the reasoning trajectories. Surprisingly, ToT is worse than CoT when using GPT-3.5. When prompted with an incomplete reasoning trajectory (i.e. some sub-questions are answered, some not asked yet), the GPT-3.5 model tends to give a final answer to the original question instead of generating the next sub-question⁵. 3) CoT-SC is slightly better than ToT. While ToT does not improve over CoT, CoT-SC can still bring some improvement through variance reduction.

Comparison with GSM8K From our analysis, we see that math word problem (GSM8K) and multi-hop question answering (MusiQue) have simple policies. However, the in-context learning performance of GPTs on GSM8K is substantially higher than that of MusiQue. We believe there are two main reasons: 1) Retrieving the appropriate information from the context to answer a sub-question, i.e. the transition, is hard in MusiQue. After manual analysis of 20 failed examples, we found that 16 of them are due to retrieval mistake, where the correct sub-question is proposed, i.e. the policy is easy, but the proposed sub-answer is incorrect. To validate this hypothesis, we did an ablation study by replacing every natural language sentence in the context with a relation triplet automatically parsed by GPT-4 with appropriate prompting. We believe this should reduce the difficulty of retrieving information. Using the parsed context improved performance substantially as shown in Figure 8b.

⁵We further explored this issue with the following experiment: instead of generating the response all the way to the end (standard CoT), stop the generation when a newline is generated. And then use the original input together with previous generated text as input to prompt the model again to get a new line. This is equivalent to the CoT, only that we are stopping every line. This method for GPT-3.5 has f1 score 41.09 indicating that GPT3.5 is not behaving as expected.

Interestingly, CoT-SC no longer improves on CoT – the variance in the problem may mostly be coming from the difficulty of retrieval and is probably substantially reduced in the parsed version. 2) Understanding the complex natural language question in MusiQue is hard. Some of the complex composed question in the dataset look like "How were the same people who the Somali Muslim Aju-ran Empire declared independence from expelled from the natural boundary between Thailand and the country where Nam Theun is found?", which can be hard for the language model to understand. From our analysis, we see that it is easy to determine which incomplete relation triplet to infer next, however, understanding the natural language and translate it into a logical expression which is a conjunction of relation triplets can be difficult. In the 20 failed examples we analyzed, 4 of them are due to wrong decomposition, i.e. the sub-question proposed is not helpful in answering the composed question.

ToT is not useful in this dataset likely because the sub-questions mostly have a single correct answer. Unlike GSM8K where each equation have only one correct answer, it is possible for sub-questions to have multiple acceptable answers that need to be refined with additional sub-questions. For datasets where this is common, ToT may be more effective. And if the knowledge graph representing the context is small, it may also be possible to linearize the search tree into a CoT of reasonable size, as demonstrated in the Air Travel Planning case study.

B.2 Blocksworld

Blocksworld is a planning task motivated by robot manipulation. It has a table with blocks in different colours. Given a set of preferred constraints for the configuration of blocks, the agent needs to output a sequence of actions to rearrange the blocks into the target state that satisfies those constraints. The agent can only pick up the block if it has no blocks on its top and place it in an open space or stack on a top block. For example, with a starting state of blocks stacking in a specific order, the goal could be "the red block is on top of the yellow block, and the blue block is at the bottom." The goal provides constraints of the goal state, and the agent should find a sequence of pick and place operations on the blocks to satisfy the constraints. It is a typical planning problem with constraint satisfaction in a finite domain, which is NP-complete to find the shortest plan (Gupta and Nau, 1992).

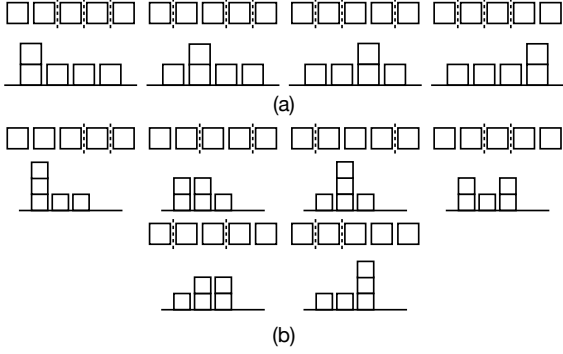


Figure 9: We take stacking 5 blocks in 4 stacks (shown in a) and 3 stacks (shown in b) as examples. (a) shows that stacking 5 blocks in 4 stacks is equivalent to choosing 3 out of 4 margins. (b) shows that stacking 5 blocks in 3 stacks is equivalent to choosing 2 out of 4 margins.

In this section, we conduct an empirical study based on Hao et al. (2024) with some new contributions: 1) We conduct finetuning experiments to verify the difficulty of learning the shortest plan solution for this problem. 2) We use a more realistic version of ToT where we replace the ground-truth action list in branches and perfect goal recognition in Hao et al. (2024) with possibly unreliable LLM-based action proposal and goal recognition. 3) We show that further decomposing the direct action proposal which does not reveal the satisfied pre-conditions into a CoT that reveals the those pre-conditions before proposing the actions improves performance.

B.2.1 Analysis

We first upper bound the number of states. Assume there are K blocks on the table. The blocks can be stacked arbitrarily on the table. Since the state is described in natural language, there must be a specific order in describing each stack of blocks. Therefore, the stacks of blocks in this state description are permutative. We first count the total number of stacking ways and their permutations, assuming the blocks are all in the same colours, i.e., without considering the order of blocks. For example, 5 blocks can be placed in 4 stacks, where one stack has two blocks while the other stacks have one block. In that case, there will be four stacking cases: (2, 1, 1, 1), (1, 2, 1, 1), (1, 1, 2, 1), and (1, 1, 1, 2). This is equivalent to choosing 3 out of 4 margins demonstrated in Figure 9 (a). 5 blocks can also be placed in 3 stacks, a choice of 2 in 4 margins, which is shown in Figure 9 (b). Thus, for K blocks, there will be $\sum_{k=0}^{K-1} \binom{K-1}{k}$ is the total number of different stacking ways. Now, when the blocks are unique in colours, there will be $K!$ different ways of arranging the order of the blocks. Thus, the total number of

states is $K! \sum_{k=0}^{K-1} \binom{K-1}{k} = K!2^{K-1}$, where $K!$ is the total number of permutation of all blocks, and $\sum_{k=0}^{K-1} \binom{K-1}{k}$ is the total number of different stacking ways. There are four types of actions available: pick up a block, unstack a block from on top of another block, put down a block, or stack a block on top of another block. The goal can be up to 3 pairwise spatial relation constraints. If the state satisfies the constraints, the task is considered a success.

Direct Learning to directly predict the sequence of actions can be represented as a table. The rows and columns of the table are the starting states and goal state constraints. Each entry records a sequence of actions. Assume the maximum number of actions is proportional to the number of blocks. The total size of the table is $O(K!K^32^{K-1} \log K)$ bits.

CoT For CoT, the actions are generated step by step. In each step, the LLM generates the next action and predicts the next state, given the action and current state. The LLM needs to learn the precondition and effect of each action. The total number of grounded actions is $2K + 2K(K - 1) = 2K^2$. Each action needs to learn its pre-conditions and effects. The state can be described using $O(K^2)$ propositions such as $\text{On}(A, B)$ and $\text{Clear}(A)$. The pre-condition is a subset of these propositions that needs to be true. The effects consist of the add effect, which is a subset of propositions that will become true upon the execution of the action, and the delete effect, which is a subset of propositions that will become false. Assuming that the subsets in the pre-condition, add effects and delete effects have size at most k , we need to describe the choice of k propositions from $O(K^2)$ possible propositions, which requires $O(k \log K)$ bits. With $O(K^2)$ actions, the total description length for describing the action preconditions and effects is $O(kK^2 \log K)$ bits. In Blocksworld, the value of k is a small constant and the complexity of the transitions seems reasonable.

Given the current state and goal constraints, we also require a policy to predict the next action. This can be represented as a look-up table whose rows and columns are the current state and goal constraints, and the entry is the next action. Describing one action requires $O(\log K)$ bits. Thus, the total size of the table is still $O(K!K^22^{K-1} \log K)$ bits, suggesting that a policy in tabular form may be difficult to learn.

ToT Unlike CoT, ToT does not require a policy for predicting the next correct actions, although it needs to learn the pre-condition and effects of

each action. The LLM needs to behave as a world model that proposes valid actions, predicts the next state, and recognizes the goal. ToT uses the same actions as CoT, and our analysis has reasonable sample complexity. Verifying whether the solution is valid takes $O(K)$ time; hence, a small representation of the goal recognizer is possible. Similar to the game of 24, the self-evaluation (heuristic) could be complex, requiring $O(K!K^22^{K-1})$ entries. However, ToT spends more computational costs to conduct look-ahead searches; hence, we expect that its performance may be better than that of CoT.

B.2.2 Experiments

We test the performance of Direct, CoT, CoT-SC, and ToT using four-block-stacking tasks. We test the in-context learning (ICL) performance of GPT-3.5 and GPT-4 and fine-tune the Llama-2-7b (Touvron et al., 2023) using Direct and CoT. We use the same testing dataset as (Hao et al., 2024)⁶. As for the fine-tuning experiments, we generate 1000 trajectories for block stacking using the same distribution of the testing data. Our in-context learning experiment is adapted from Hao et al. (2024), but we have modified the reasoning methods: 1) The CoT in Hao et al. (2024) is essentially Direct in our definition, while our CoT keeps tracking the state changes in each step. 2) Our ToT modified the RAP method in Hao et al. (2024) to use LLM to propose all the actions in the branches, predict the next state after applying one action, and recognize the goal. In comparison, the RAP only use LLM for next-state prediction and uses the ground truth admissible actions. 3) We provide two versions of ToT: ToT and ToT-Decomp. ToT uses LLM to propose possible actions directly given the current state. ToT-Decomp uses a CoT prompt to propose possible actions. Instead of proposing possible actions directly from the current state, ToT-Decomp first explains the reasons that each proposed action is admissible (pre-conditions satisfied) before generating the possible action branches. For example, ToT-Decomp would explain that Holding(x) and Clear(y) are both be true before proposing Stack(x, y) as a possible action. This can also be viewed as making the pre-conditions that are hidden in the direct prompt visible via CoT prompting.

The experimental results are reported in the Figure 10. Both the Direct and CoT do not perform well in the fine-tuning experiments. It suggests that learning the short-chain solution for this planning

⁶<https://github.com/matrix-org/llm-reasoners>

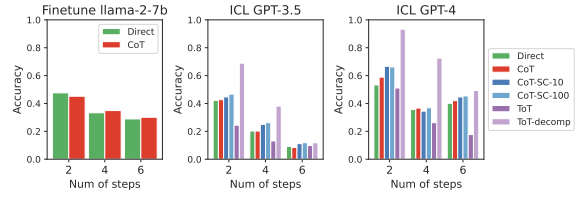


Figure 10: Results of the Blocksworld experiments. ICL stands for in-context learning.

problem is not sample-efficient. This is because finding the short-chain solution for this problem is likely computationally hard. It is also reflected by the description length analysis: Direct and CoT have similar, very large tabular description lengths, suggesting that their sample complexities are both very high.

For the in-context learning (ICL), we also found that Direct and CoT have very similar performance in both experiments using GPT-3.5 and GPT-4. When using CoT-SC with 10 samples and 100 samples, the performance improved a bit in general, but not very much. CoT-SC achieves higher performance than CoT because it spends more on computation by sampling more reasoning trajectories and performing a majority vote, effectively decreasing the variance and improving the performance. Simple ToT performs even worse than Direct. This is because the LLM was not able to learn the pre-conditions directly from the small number of examples and the invalid actions compromised the entire performance. On the other hand, using ToT-Decomp achieved the best performance. This is because using a reasonable decomposition which makes the hidden pre-conditions of the actions visible produced actions that are almost all admissible, enabling the lookahead search of ToT to be effective. Even though the self-evaluation is inaccurate, ToT searches larger areas using the more accurate reward model and transitions, making it more effective than CoT and its variants.

C GSM8K

C.1 Statistics

In the GSM8K dataset, the variable values (K) are usually from 1 to 6 digits and the average number of variables per question (N) is 3.93. The average number of variables (a_i) is 2.19, the average number of reasoning steps is 3.17, so each step is relatively simple and the number of steps are also small. We also find that the number of rules that need to be learned as world knowledge appears to be relatively small (see Appendix C.2). Overall, we see that

the decomposed components in CoT are relatively simple, and suggest that decomposition with CoT may be reasonable for GSM problems.

C.2 Common Rules in GSM8K

We analyzed 50 problems from the GSM8K training set and identified a set of rules. The first five are general rules that can be inferred from the questions and are applicable to multiple problems. The last four are question-specific rules, involving commonsense knowledge that are not mentioned in the questions.

1. Amount A = Amount B * multiplier
2. Amount A = Amount B + difference
3. Total = N_unit * Amount per unit
4. Total = Sum of components
5. Current Amount = Initial Amount - Amount Given + Amount Received
6. Question-specific (implicit): One hour = 60 Minutes
7. Question-specific (implicit): one sandwich has two slices of bread
8. Question-specific (implicit): face has two eyes
9. Question-specific (implicit): 1 quarter = 25 cent; 1 dime = 10 cent; 1 nickel = 5 cent

C.3 An Efficient Algorithm for GSM8K

Based on our analysis of the GSM8K problems in 4.1.1, we give a formulation of the GSM8K problems, and show that there exists an algorithm that has runtime linear to the total input length.

Problem Formulation

Input: A set of N variables $\{V_1, \dots, V_N\}$, where the values of some variables are known (from natural language input), while some are unknown (to be inferred); A set of M equations $\{R_1, \dots, R_M\}$, where all equations have exactly one variable on LHS; A target variable V_t whose value we want to know.

Output: The value of V_t .

The solvability of the problem ensures that for all variables, if the value is not given in the natural language question, will appear on the LHS of some equation.

An Efficient Algorithm :

Inspired by Dowling and Gallier (1984), we design an algorithm whose runtime is linear to the size of the problem (total length of all equations).

We maintain a list $\text{numvars}[M]$ which stores the number of unsolved variables on RHS for each equation; a list $\text{lhslist}[M]$ which stores which variable is on LHS of an equation; a list $\text{equationlist}[N]$ which stores the index of the equations where the corresponding variable appears on RHS. We say an equation R_i is ready to be processed if $\text{numvars}[i] = 0$. We maintain a queue that will contain the equations that are ready to be processed, and it is initialized to contain the equations that are ready to process given the known variables from natural language input.

Then we loop over the queue. Let equation1 be the current head of the queue and let $\text{nextvar}=\text{lhslist}[\text{equation1}]$ be the variable on the LHS of of equation1 . Pop the head of the queue, and iterate over $\text{equationlist}[\text{nextvar}]$, for every equation2 in it, reduce $\text{numvars}[\text{equation2}]$ by 1, and if $\text{numvars}[\text{equation2}]$ becomes 0, add equation2 to the queue.

Loop until the queue is empty, we would have solved the values of all N variables. Refer to Algorithm 1 for a more concise representation of the algorithm.

Complexity of the Algorithm numvars and lhslist can be initialized in $O(L)$, where L is the total length of all equations. When processing an equation, the decrement of numvars corresponds to the "deletion" of occurrences of the variable in an equation, each variable in the equation is looked only once, thus processing all equations also runs in $O(L)$. Overall, the runtime of the algorithm is $O(L)$, i.e. linear to the total length of the equations. If we assume each variable appears only once on the LHS of equations, and each equation has a number of variables up to a constant C (as we have seen in 4.1.1, this is often true for GSM8K problems), the runtime of the algorithm would be $O(N)$, i.e. linear to the number of variables in the problem.

C.4 GSM8K Errors

We observe three types of errors from GPT, namely grounding error, calculation error, and missing rule. We provide an example for each type.

Listing 1: Grounding Error

Question:
Annie was given a pack of crayons. The pack contained 21 crayons. She already had a box of 36 crayons in her locker.

Her friend Bobby gave her half the amount she already had in her locker. She decided to give her sister Mary 1/3 of her total amount of crayons. How many crayons does she give to Mary? 1438
 1439
 1440
 1441
 1442
 GPT-3.5 response: 1443
 "Annie originally had 21 crayons in the pack and 36 crayons in her locker. 1444
 So she had a total of $21 + 36 = 57$ crayons. 1445
 Bobby gave her half the amount she already had, so he gave her $57 / 2 = 28.5$ crayons, but since you can't have half a crayon, let's round down to 28. 1446
 So she now has $57 + 28 = 85$ crayons. 1447
 So 1/3 of her total amount of crayons is $85 / 3 = 28.33$ crayons, but since you can't have a fraction of a crayon, let's round down to 28 crayons. 1448
 So she gives Mary 28 crayons. 1449
 So the answer is 28." 1450
 1451
 1452
 1453
 1454
 1455

Algorithm 1 Forward Chaining Algorithm for GSM8K Problems

Require:

- N variables $\{V_1, \dots, V_N\}$, with known and unknown values.
- M equations $\{R_1, \dots, R_M\}$, each with one variable on LHS.
- Target variable V_t to find the value of.

Ensure:

Value of V_t .

- 1: $\text{numvars}[1 \dots M] \leftarrow$ Number of unsolved variables on RHS of each R_i .
 - 2: $\text{lhslist}[1 \dots M] \leftarrow$ Variable on LHS of each R_i .
 - 3: $\text{equationlist}[1 \dots N] \leftarrow$ Indices of R_i where V_j appears on RHS.
 - 4: Initialize queue readyQueue with indices of R_i where $\text{numvars}[i] = 0$.
 - 5: **while** readyQueue is not empty **do**
 - 6: $i \leftarrow$ Dequeue from readyQueue .
 - 7: $V_j \leftarrow \text{lhslist}[i]$.
 - 8: **for each** k in $\text{equationlist}[j]$ **do**
 - 9: $\text{numvars}[k] \leftarrow \text{numvars}[k] - 1$.
 - 10: **if** $\text{numvars}[k] = 0$ **then**
 - 11: Enqueue k into readyQueue .
 - 12: **end if**
 - 13: **end for**
 - 14: **end while**
 - 15: **return** Value of V_t .
-

In Listing 1, the problem involves using the rule "Amount $A = \text{Amount } B * \text{multiplier}$ ", which should be grounded as "Amount from Bobby = Amount in Locker * 1/2", not "Amount from Bobby = Amount Annie have in total * 1/2". 1456
 1457
 1458
 1459
 1460

Listing 2: Calculation Error

Question: 1461
 To get free delivery, Alice needs to spend a minimum of \$35.00 online at her favorite grocery store. In her cart she has 1.5 pounds of chicken at \$6.00 per pound, 1 pack of lettuce for \$3.00, cherry tomatoes for \$2.50, 4 sweet potatoes at \$0.75 each, 2 heads of broccoli for \$2.00 each and a pound of Brussel sprouts for \$2.50. How much more does she need to spend in order to get free delivery? 1462
 1463
 1464
 1465
 1466
 1467
 1468
 1469
 GPT-3.5 response: 1470
 "Alice's total so far is: 1471
 $(1.5 * 6) + 3 + 2.50 + (4 * 0.75) + (2 * 2) + 2.50$ 1472
 $= 9 + 3 + 2.50 + 3 + 4 + 2 + 2.50$ 1473
 $= 26.00$ 1474
 1475
 To get free delivery, she needs to spend at least \$35.00. 1476
 So she needs to spend $35 - 26$ 1477
 $= 9.00$ more to get free delivery. 1478
 1479
 So the answer is 9.00." 1480

In Listing 2, the calculation is wrong. 1481

Listing 3: Missing Rule

Question: 1482
 A Ferris wheel can accommodate 70 people in 20 minutes. If the Ferris wheel is open from 1:00 pm until 7:00 pm, how many people will get to ride? 1483
 1484
 1485
 1486
 GPT-3.5 response: 1487
 "The Ferris wheel can accommodate 70 people in 20 minutes. 1488
 There are 6 periods of 20 minutes from 1:00 pm to 7:00 pm. 1489
 So in total, the Ferris wheel can accommodate $70 * 6 = 420$ people. 1490
 So 420 people will get to ride the Ferris wheel from 1:00 pm until 7:00 pm. 1491
 So the answer is 420." 1492
 1493
 1494

In Listing 3, one commonsense knowledge (one hour has 60 minutes) not mentioned in the question is not properly used in the solution. 1495
 1496
 1497

D Dynamic Programming 1498

D.1 Algorithm for MWIS 1499

An algorithm that solves the Maximum Weighted Independent Set problem and runs in time $O(N)$ where N is the number of numbers in the input is shown in Algorithm 2. 1500
 1501
 1502
 1503

Algorithm 2 Dynamic Programming Algorithm for the Maximum Weighted Independent Set problem

Require: An array arr of integers
Ensure: A sequence of decisions maximizing a certain criterion based on arr

```
1:  $N \leftarrow$  length of  $arr$ 
2: Initialize  $dp[0 \dots N - 1]$  with zeros
3:  $dp[N - 1] \leftarrow \max(arr[N - 1], 0)$ 
4:  $dp[N - 2] \leftarrow \max(arr[N - 1], arr[N - 2], 0)$ 
5: for  $i \leftarrow N - 3$  downto 0 do
6:    $dp[i] \leftarrow \max(dp[i + 1], arr[i] + dp[i + 2], 0)$ 
7: end for
8: Initialize  $result$  as an empty list
9:  $can\_access\_next\_item \leftarrow \mathbf{true}$ 
10: for  $i \leftarrow 0$  to  $N - 3$  do
11:   if  $dp[i] = arr[i] + dp[i + 2]$  and  $can\_access\_next\_item$  then
12:     Append 1 to  $result$ 
13:      $can\_access\_next\_item \leftarrow \mathbf{false}$ 
14:   else
15:     Append 2 to  $result$ 
16:      $can\_access\_next\_item \leftarrow \mathbf{true}$ 
17:   end if
18: end for
19: if  $dp[N - 2] = arr[N - 2]$  and  $can\_access\_next\_item$  then
20:   Append 1 to  $result$ 
21: else
22:   Append 2 to  $result$ 
23: end if
24: if  $dp[N - 1] = arr[N - 1]$  and  $can\_access\_next\_item$  then
25:   Append 1 to  $result$ 
26: else
27:   Append 2 to  $result$ 
28: end if
29: return  $result$ 
```

E Inductive bias of Transformers

From the math word problem and dynamic programming fine-tuning experiments, we see that for some tasks (maximum weighted independent set), the transformer can learn to directly answer the problem efficiently, while for some other tasks (word problem), the direct answer is hard to learn.

We conduct two more experiments to study what might affect the performance of learning to direct answer other than sample complexity: 1) learn the max function, where the input is a list of integers, and the expected output is the maximum value in the input list. This requires only looping over the sequence once, and storing one intermediate value; 2) another dynamic programming problem called rain water⁷ that requires looping over the array three times and storing two one-dimensional arrays for memorization. These two problems are similar to MWIS as they all require looping over the input sequence and maintaining some internal variables during the iteration. We use them to study whether the difference between learning to directly answer the word problem and MWIS is related to the inductive bias of transformers. To eliminate the confounding part, the difficulty of language in the word problem, we perform a modified version of the problem, where we remove all natural language in the prompt, and use a fixed formula for ground-truth answer: $(v_1v_2 + v_1v_3 + v_1v_3/v_5 + v_1v_2/v_4)v_7/v_6$. The input would look like "1, 6, 4, 3, 2, 14, 8", and the expected output for this example would be "8" $((1 \cdot 6 + 1 \cdot 4 + 1 \cdot 4/2 + 1 \cdot 6/3) \cdot 8/14 = 8)$. We randomly sample the values of the variables, ensuring the answer value is integer to construct the dataset.

Task	Accuracy (%)
MWP	58.00
MWIS ($n \in [4, 5, 6]$)	98.89
MWIS ($n = 200$)	0.01
max ($n = 30$)	99.50
rain water ($n = 10$)	89.00

Table 1: Fine-tuning results of different problems. MWP stands for the modified word problem where the input contains only 7 numbers. All tasks are fine-tuned with 10k direct answer examples and evaluated on in-domain examples.

From the table, we see that MWIS, max, and rain

⁷<https://leetcode.com/problems/trapping-rain-water/>

water perform significantly better than MWP. This suggests that it might be easy for transformers to learn this loop type of problem when the problem size is small. However, when the problem size of MWIS is large ($n = 200$), the model fails to generalize to unseen test examples. This aligns with previous findings (Weiss et al., 2021; Zhou et al., 2024) that suggest that it would consume one transformer layer to approximate one iteration in an algorithm. And with a problem size of 200, it can be hard for transformers to approximate the algorithm in a generalizable way, thus some other patterns in the training set may be exploited, leading to poor generalization.

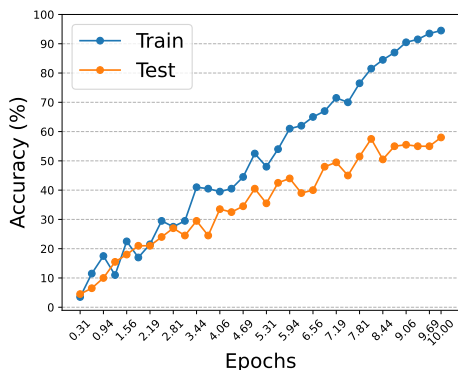


Figure 11: Results of fine-tuning word problem.

From Fig. 11 we can see that for the modified word problem, the transformer can fit the training set reasonably well, while the test set performance peaks at 58.0%. This suggests that by learning to answer directly, the transformer is behaving similarly to learning by filling a table, instead of learning the underlying rational function, which supports our description length analysis.

F Air travel planning

We use the Kaggle World Cities⁸ database data and sample 212 cities with more than 1 million populations. We sampled 58 large cities and 154 mid-sized cities. We use the Virtual Radar Server⁹ to get the real-time (Jan 13, 2024) flight data as the ground truth.

G Game of 24

We test the hard games indexed 901-1000 from 4nums.com. An output is considered correct if the expression evaluates to 24 and uses all input numbers once.

⁸<https://www.kaggle.com/datasets/max-mind/world-cities-database>

⁹<https://github.com/vradarserver/standing-data>

H Tables

Method	GPT-3.5	GPT-4
Direct	28.51	47.16
CoT	79.53	94.09
ToT	81.88	96.00

Table 2: Figure 2a

Dataset size	Direct	CoT
1000	18.50	88.00
2000	22.50	88.00
3000	30.50	92.50
4000	35.00	93.50
5000	37.50	95.00
6000	46.50	95.00
7000	46.50	96.00
8000	48.50	96.50
9000	48.50	97.50
10000	58.00	96.50

Table 3: Figure 2b

Method	InD	OoD	Total
Direct 0s	38.67	8.75	21.57
Direct 3s	68.67	35.25	49.57
Direct 6s	57.67	30.25	42.00
CoT 0s	59.33	35.25	45.57
CoT Implicit 3s	67.00	41.50	52.43
CoT Implicit 6s	65.00	36.25	48.57
CoT Explicit 3s	85.67	65.00	73.86
CoT Explicit 6s	86.67	66.50	75.14

Table 4: Figure 3 GPT-4

Method	InD	OoD	Total
Direct 0s	26.00	13.50	18.86
Direct 3s	32.33	10.50	19.86
Direct 6s	39.33	20.25	28.43
CoT 0s	24.33	8.75	15.43
CoT Implicit 3s	18.00	6.75	11.57
CoT Implicit 6s	20.33	5.00	11.57
CoT Explicit 3s	56.67	16.00	33.43
CoT Explicit 6s	63.33	28.75	43.57

Table 5: Figure 3 GPT-3.5

Method	Large cities	Mid-sized cities
CoT 0s	70.76	50.00
CoT 3s	73.10	51.64
CoT 8s	72.51	53.27
CoT-SC 0s	72.51	55.47
CoT-SC 3s	74.56	54.13
CoT-SC 8s	75.43	56.85
ToT-linear 0s	75.43	69.67
ToT-linear 3s	81.29	77.05
ToT-linear 8s	78.36	72.95
ToT 0s	78.36	72.13
ToT 3s	80.70	75.41
ToT 8s	81.29	75.41

Table 6: Figure 5 GPT-3.5 (Accuracy, %)

Method	Large cities	Mid-sized cities
CoT 0s	71.35	64.75
CoT 3s	76.02	68.03
CoT 8s	85.38	70.49
CoT-SC 0s	74.43	66.21
CoT-SC 3s	79.59	69.67
CoT-SC 8s	87.13	71.12
ToT-linear 0s	54.24	47.54
ToT-linear 3s	87.13	69.67
ToT-linear 8s	84.80	68.85
ToT 0s	76.02	70.49
ToT 3s	88.30	78.69
ToT 8s	88.89	79.51

Table 7: Figure 5 GPT-4 (Accuracy, %)

Num of edges	Large cities	Mid-sized cities
1069	90.64±2.21	80.32±3.21
2138	93.30±2.02	85.87±3.92
4277	97.07±0.94	90.16±1.45
6415	97.90±1.20	93.79±1.13

Table 8: Figure 6 ToT-linear (Accuracy % ± standard error)

Num of edges	Large cities	Mid-sized cities
744	65.50±5.22	58.10±4.91
1489	78.94±3.90	68.85±4.56
2979	80.19±4.12	74.59±4.11
4468	81.52±5.23	77.97±5.10
5958	83.04±3.54	81.98±3.41

Table 9: Figure 6 CoT (Accuracy % ± standard error)

Method	GPT-4	GPT-3.5
ToT 5s	58	20
ToT-Decomp 5s	86	47
ToT-Decomp 3s	23	20
ToT-Decomp 1s	19	15
CoT 5s	6	2
CoT-SC 5s	11	7
Direct 5s	10	4

Table 10: Figure 7, main results (Accuracy, %).

Method	Transition error	Proposal error
TOT-GPT4-5s	7.12	2.04
TOT-GPT4-Decomp-5s	2.80	1.44
TOT-GPT3.5-5s	16.62	3.15
TOT-GPT3.5-Decomp-5s	3.06	0.30

Method	Missing action	Answer error
TOT-GPT4-5s	12.44	10.04
TOT-GPT4-Decomp-5s	6.63	1.56
TOT-GPT3.5-5s	23.63	19.03
TOT-GPT3.5-Decomp-5s	16.60	2.28

Table 11: Figure 7, main results (Error rate, %).

Method	GPT-4	GPT-3.5
Direct	62.5	42.14
Direct (parsed)	73.7	53.74
CoT	64.33	52.91
CoT (parsed)	76.89	65.62
CoT-SC	66.73	53.79
CoT-SC (parsed)	75.89	65.67
ToT	65.36	45.68
ToT (parsed)	75.72	49.98

Table 12: Figure 8

Method	2-step	4-step	6-step
Direct	47.6	33.2	28.9
CoT	45.2	34.9	30.1

Table 13: Figure 10a, results of finetuning Llama-2-7b (Accuracy, %).

Method	2-step	4-step	6-step
Direct	42.2	20.2	9.2
CoT	42.7	20.2	8.4
CoT-SC-10	44.7	25.0	11.2
CoT-SC-100	46.7	26.2	11.8
ToT	22.4	13.1	9.8
ToT-Decomp	68.9	38.1	11.8

Table 14: Figure 10b, results of GPT-3.5 for in-context learning (Accuracy, %).

Method	2-step	4-step	6-step
Direct	53.3	35.7	40.1
CoT	58.9	36.7	42.0
CoT-SC-10	66.7	34.5	44.7
CoT-SC-100	66.2	36.9	45.4
ToT	51.1	26.2	11.7
ToT-Decomp	93.3	72.6	49.3

Table 15: Figure 10c, results of GPT-4 for in-context learning (Accuracy, %).

I Prompts

I.1 GSM8K Prompts

Listing 4: GSM8K Direct prompt

```

direct_8s = """Please answer a math word problem given the
following exampls. Respond only the answer, in the format "
The answer is ###."
Example:
Question: There are 15 trees in the grove. Grove workers will
plant trees in the grove today. After they are done, there
will be 21 trees. How many trees did the grove workers plant
today?
The answer is 6.

Question: If there are 3 cars in the parking lot and 2 more
cars arrive, how many cars are in the parking lot?
The answer is 5.

Question: Leah had 32 chocolates and her sister had 10 more
chocolates than her. If they ate 35, how many pieces do they
have left in total?
The answer is 39.

Question: Jason had 20 lollipops. He gave Denny some lollipops.
Now Jason has 12 lollipops. How many lollipops did Jason give
to Denny?
The answer is 8.

Question: Shawn has five toys. For Christmas, he got two toys
each from his mom and dad. How many toys does he have now?
The answer is 9.

Question: There were nine computers in the server room. Five
more computers were installed each day, from monday to
thursday. How many computers are now in the server room?
The answer is 29.

Question: Michael had 58 golf balls. On tuesday, he lost 23
golf balls. On wednesday, he lost 2 more. How many golf balls
did he have at the end of wednesday?
The answer is 33.

Question: Olivia has $23. She bought five bagels for $3 each.
How much money does she have left?
The answer is 8.

Question: {question}

```

```

The answer is
"""

```

Listing 5: GSM8K CoT and ToT prompt

```

cot_8s = """Please answer a math word problem given the
following example. Respond with reasoning steps, and end with
the answer, in the format "So the answer is ###."
Example:
Let's think step by step.
Question: There are 15 trees in the grove. Grove workers will
plant trees in the grove today. After they are done, there
will be 21 trees. How many trees did the grove workers plant
today?
Solution: There are 15 trees originally.
And there were 21 trees after some more were planted.
So 21 - 15 = 6 trees were planted.
So the answer is 6.

Let's think step by step.
Question: If there are 3 cars in the parking lot and 2 more
cars arrive, how many cars are in the parking lot?
Solution: There are originally 3 cars.
And 2 more cars arrive.
So there are 3 + 2 = 5 cars now.
So the answer is 5.

Let's think step by step.
Question: Leah had 32 chocolates and her sister had 10 more
chocolates than her. If they ate 35, how many pieces do they
have left in total?
Solution: Originally, Leah had 32 chocolates.
And her sister had 10 more chocolates than her.
So her sister had 42 chocolates.
So in total they had 32 + 42 = 74 chocolates.
Then they ate 35 chocolates.
Therefore they had 74 - 35 = 39 chocolates left.
So the answer is 39.

Let's think step by step.
Question: Jason had 20 lollipops. He gave Denny some lollipops.
Now Jason has 12 lollipops. How many lollipops did Jason give
to Denny?
Solution: Jason started with 20 lollipops.
Then he had 12 after giving some to Denny.
So he gave Denny 20 - 12 = 8 lollipops.
So the answer is 8.

Let's think step by step.
Question: Shawn has five toys. For Christmas, he got two toys
each from his mom and dad. How many toys does he have now?
Solution: Shawn started with 5 toys.
And he got 2 toys each from his mom and dad.
So he got 2 + 2 = 4 toys.
Therefore, he has 5 + 4 = 9 toys now.
So the answer is 9.

Let's think step by step.
Question: There were nine computers in the server room. Five
more computers were installed each day, from monday to
thursday. How many computers are now in the server room?
Solution: There were originally 9 computers.
And 5 more computers were added from onday to thursday.
There are 4 days between monday and thursday.
So 5 * 4 = 20 computers were added in total.
So there are 9 + 20 = 29 computers now.
So the answer is 29.

Let's think step by step.
Question: Michael had 58 golf balls. On tuesday, he lost 23
golf balls. On wednesday, he lost 2 more. How many golf balls
did he have at the end of wednesday?
Solution: Michael started with 58 golf balls.
And he lost 23 golf balls on tuesday.
So after losing 23 on tuesday, he had 58 - 23 = 35.
And then he lost 2 more golf balls on wednesday.
So after losing 2 more on wednesday, he had 35 - 2 = 33 golf
balls.
So the answer is 33.

Let's think step by step.
Question: Olivia has $23. She bought five bagels for $3 each.
How much money does she have left?
Solution: Olivia had 23 dollars.
And she bought 5 bagels.
And each bagel costs 3 dollars.

```

1703 So she spent $5 * 3 = 15$ dollars.
 1704 So she has $23 - 15 = 8$ dollars left.
 1705 So the answer is 8.
 1706
 1707 Let's think step by step.
 1708 Question: {question}
 1709 Solution:
 1710 ""
 1711

Listing 6: GSM8K ToT self-evaluation prompts

1711 evaluate_prompt = ''
 1712 Q: Julie climbed 15 steps up the giant slide. She climbed down
 1713 6 steps to talk to her friend, Maria. Then she climbed up 8
 1714 steps to get to the top. How many steps does the slide have?
 1715
 1716 A:
 1717 Julie climbed 15 steps up.
 1718 # Is the above step of reasoning:
 1719 # (A) Correct
 1720 # (B) Incorrect
 1721 # The above step of reasoning is (A)
 1722 Then she climbed down 6 steps.
 1723 # Is the above step of reasoning:
 1724 # (A) Correct
 1725 # (B) Incorrect
 1726 # The above step of reasoning is (A)
 1727 Then she climbed up 8 steps.
 1728 # Is the above step of reasoning:
 1729 # (A) Correct
 1730 # (B) Incorrect
 1731 # The above step of reasoning is (A)
 1732 So she climbed $15 + 8 = 23$ steps.
 1733 # Is the above step of reasoning:
 1734 # (A) Correct
 1735 # (B) Incorrect
 1736 # The above step of reasoning is (B), because she also climbed
 1737 down 6 steps, so she climbed $23 - 6 = 17$ steps.
 1738 So the slide has 23 steps.
 1739 # Is the above step of reasoning:
 1740 # (A) Correct
 1741 # (B) Incorrect
 1742 # The above step of reasoning is (A), but the value of steps
 1743 of slides is incorrect.
 1744 So the answer is 23.
 1745 # Is the above step of reasoning:
 1746 # (A) Correct
 1747 # (B) Incorrect
 1748 # The above step of reasoning is (A), but the value of steps
 1749 of slides is incorrect.
 1750
 1751
 1752
 1753
 1754
 1755 Q: Suzanne read the first 15 pages of her book on Monday. She
 1756 read 16 more pages than that on Tuesday. Then there were 18
 1757 pages left. How many pages are in Suzanne's book altogether?
 1758
 1759 A:
 1760 Suzanne read 15 pages on Monday.
 1761 # Is the above step of reasoning:
 1762 # (A) Correct
 1763 # (B) Incorrect
 1764 # The above step of reasoning is (A)
 1765 Then she read 16 more pages on Tuesday.
 1766 # Is the above step of reasoning:
 1767 # (A) Correct
 1768 # (B) Incorrect
 1769 # The above step of reasoning is (A)
 1770 So she read $15 + 16 = 31$ pages in total.
 1771 # Is the above step of reasoning:
 1772 # (A) Correct
 1773 # (B) Incorrect
 1774 # The above step of reasoning is (B), because she read 16 more
 1775 pages than that on Tuesday, so she read $15 + 16 = 31$ pages on
 1776 tuesday. So she read $15 + 31 = 46$ pages in total.
 1777 Then there were 18 pages left.
 1778 # Is the above step of reasoning:
 1779 # (A) Correct
 1780 # (B) Incorrect
 1781 # The above step of reasoning is (A), but the value of total
 1782 read pages of monday and tuesday is incorrect.
 1783 So the book had $31 + 18 = 49$ pages.
 1784 # Is the above step of reasoning:
 1785 # (A) Correct

(B) Incorrect
 # The above step of reasoning is (A), but the value of total
 read pages of monday and tuesday is incorrect. So the book had
 $46 + 18 = 64$ pages.
 So the answer is 49.
 # Is the above step of reasoning:
 # (A) Correct
 # (B) Incorrect
 # The above step of reasoning is (A), but the value of total
 read pages of monday and tuesday is incorrect.
 1786
 1787
 1788
 1789
 1790
 1791
 1792
 1793
 1794
 1795
 1796
 1797
 1798
 1799
 1800
 1801 Q: Allison brought some CDs online. Each CD cost \$7. There was
 1802 an additional charge of \$4 per order for shipping costs. The
 1803 total bill came to \$60. How many CDs did Allison buy?
 1804
 1805 A:
 1806 Each CD cost 7 dollars.
 1807 # Is the above step of reasoning:
 1808 # (A) Correct
 1809 # (B) Incorrect
 1810 # The above step of reasoning is (A)
 1811 And there was an additional charge of 4 dollars.
 1812 # Is the above step of reasoning:
 1813 # (A) Correct
 1814 # (B) Incorrect
 1815 # The above step of reasoning is (A)
 1816 So the total cost of each CD is $7 + 4 = 11$ dollars.
 1817 # Is the above step of reasoning:
 1818 # (A) Correct
 1819 # (B) Incorrect
 1820 # The above step of reasoning is (B), because each CD cose 7
 1821 dollars.
 1822 So $60 / 11 = 5.45$.
 1823 # Is the above step of reasoning:
 1824 # (A) Correct
 1825 # (B) Incorrect
 1826 # The above step of reasoning is (B), because it cost 4
 1827 dollars for shipping costs. So the cost of CDs is $60 - 4 = 56$
 1828 dollars. So Allison bought $56 / 7 = 8$ CDs.
 1829 So the answer is 5.
 1830 # Is the above step of reasoning:
 1831 # (A) Correct
 1832 # (B) Incorrect
 1833 # The above step of reasoning is (A), but the value of number
 1834 of CDs is incorrect.
 1835
 1836
 1837
 1838
 1839
 1840 Q: Luis and Cameron shared some stickers is the ratio 5:2.
 1841 Luis received 15 more stickers than Cameron. How many stickers
 1842 were there altogether?
 1843
 1844 A:
 1845 Let's say there were x stickers.
 1846 # Is the above step of reasoning:
 1847 # (A) Correct
 1848 # (B) Incorrect
 1849 # The above step of reasoning is (A)
 1850 Then Luis got $5x/7$ and Cameron got $2x/7$.
 1851 # Is the above step of reasoning:
 1852 # (A) Correct
 1853 # (B) Incorrect
 1854 # The above step of reasoning is (A)
 1855 Luis got 15 more than Cameron, so $5x/7 - 2x/7 = 15$.
 1856 # Is the above step of reasoning:
 1857 # (A) Correct
 1858 # (B) Incorrect
 1859 # The above step of reasoning is (A)
 1860 So $3x/7 = 15$.
 1861 # Is the above step of reasoning:
 1862 # (A) Correct
 1863 # (B) Incorrect
 1864 # The above step of reasoning is (A)
 1865 So $x = 105$.
 1866 # Is the above step of reasoning:
 1867 # (A) Correct
 1868 # (B) Incorrect
 1869 # The above step of reasoning is (B), because $3x/7 = 15$. So x
 1870 = $15 * 7 / 3 = 35$. So there were 35 stickers.
 1871 So the answer is 105.
 1872 # Is the above step of reasoning:

```

1873 # (A) Correct
1874 # (B) Incorrect
1875 # The above step of reasoning is (A), but the value of number
1876 of stickers is incorrect.
1877
1878
1879
1880
1881
1882 Q: Alexa has 92 cents in her pocket. She wants to buy 3
1883 pencils at the school supply store. Each pencil costs 8 cents.
1884 How much money will Alexa have left?
1885
1886 A:
1887 Alexa has 92 cents.
1888 # Is the above step of reasoning:
1889 # (A) Correct
1890 # (B) Incorrect
1891 # The above step of reasoning is (A)
1892 And 3 pencils for 8 cents each will be  $3 * 8 = 24$  cents.
1893 # Is the above step of reasoning:
1894 # (A) Correct
1895 # (B) Incorrect
1896 # The above step of reasoning is (A)
1897 So she has  $92 - 24 = 68$  cents left.
1898 # Is the above step of reasoning:
1899 # (A) Correct
1900 # (B) Incorrect
1901 # The above step of reasoning is (A)
1902 So the answer is 68.
1903 # Is the above step of reasoning:
1904 # (A) Correct
1905 # (B) Incorrect
1906 # The above step of reasoning is (A)
1907
1908
1909
1910
1911
1912
1913 Q: {input}
1914
1915 A: {output}
1916 # Is the above step of reasoning:
1917 # (A) Correct
1918 # (B) Incorrect
1919 # The above step of reasoning is ''

```

I.2 MWIS Prompts

Listing 7: Direct prompts

```

1921 direct_0s = """Given a sequence of integers, find a
1922 subsequence with the highest sum, such that no two numbers in
1923 the subsequence are adjacent in the original sequence.
1924
1925 To indicate the selected numbers, print an array with "1" for
1926 chosen numbers and "2" for unchosen ones. For instance, [1, 2,
1927 2, 2, 2] implies selecting only the first number. If multiple
1928 solutions exist, select the lexicographically smallest.
1929
1930
1931 {prompt}
1932 """
1933
1934 direct_3s = """Given a sequence of integers, find a
1935 subsequence with the highest sum, such that no two numbers in
1936 the subsequence are adjacent in the original sequence.
1937
1938 To indicate the selected numbers, print an array with "1" for
1939 chosen numbers and "2" for unchosen ones. For instance, [1, 2,
1940 2, 2, 2] implies selecting only the first number. If multiple
1941 solutions exist, select the lexicographically smallest.
1942
1943
1944 Let's solve input = [1, 1, -5, -1].
1945 Answer: [1, 2, 2, 2]
1946
1947
1948 Let's solve input = [3, 2, 1, -1, 2].
1949 Answer: [1, 2, 1, 2, 1]
1950
1951
1952 Let's solve input = [0, 4, -2, 3, -3, -1].
1953 Answer: [2, 1, 2, 1, 2, 2]

```

```

{prompt}
"""

direct_6s = """Given a sequence of integers, find a
subsequence with the highest sum, such that no two numbers in
the subsequence are adjacent in the original sequence.

To indicate the selected numbers, print an array with "1" for
chosen numbers and "2" for unchosen ones. For instance, [1, 2,
2, 2, 2] implies selecting only the first number. If multiple
solutions exist, select the lexicographically smallest.

Let's solve input = [1, 1, -5, -1].
Answer: [1, 2, 2, 2]

Let's solve input = [3, 2, 1, -1, 2].
Answer: [1, 2, 1, 2, 1]

Let's solve input = [0, 4, -2, 3, -3, -1].
Answer: [2, 1, 2, 1, 2, 2]

Let's solve input = [-3, -4, 4, -1]
Answer: [2, 2, 1, 2]

Let's solve input = [3, 4, -3, -1, -4]
Answer: [2, 1, 2, 2, 2]

Let's solve input = [-4, 5, 0, 2, 3, -4]
Answer: [2, 1, 2, 2, 1, 2]

{prompt}
"""

```

Listing 8: CoT Implicit prompts

```

cot_implicit_3s = """Given a sequence of integers, find a
subsequence with the highest sum, such that no two numbers in
the subsequence are adjacent in the original sequence.

To indicate the selected numbers, print an array with "1" for
chosen numbers and "2" for unchosen ones. For instance, [1, 2,
2, 2, 2] implies selecting only the first number. If multiple
solutions exist, select the lexicographically smallest.

We will solve any task instance by using dynamic programming.
We define dp[i] as the maximum sum of a subsequence that does
not include adjacent elements, when considering only the
elements of the input from the i-th position onwards.

Let's solve input = [1, 1, -5, -1].

dp[3] = max(input[3], 0) = max(-1, 0) = 0
dp[2] = max(input[2], input[3], 0) = max(-5, -1, 0) = 0
dp[1] = max(dp[2], input[1] + dp[3], 0) = max(0, 1 + 0, 0) = 1
dp[0] = max(dp[1], input[0] + dp[2], 0) = max(1, 1 + 0, 0) = 1

Finally, we reconstruct the lexicographically smallest
subsequence that fulfills the task objective by selecting
numbers as follows. We store the result on a list named "
output".

Let can_use_next_item = True.
Since dp[0] == input[0] + dp[2] (1 == 1 + 0) and
can_use_next_item == True, we store output[0] = 1. We update
can_use_next_item = False.
Since dp[1] != input[1] + dp[3] (1 != 1 + 0) or
can_use_next_item == False, we store output[1] = 2. We update
can_use_next_item = True.
Since dp[2] != input[2] (0 != -5) or can_use_next_item ==
False, we store output[2] = 2. We update can_use_next_item =
True.
Since dp[3] != input[3] (0 != -1) or can_use_next_item ==
False, we store output[3] = 2.

Reconstructing all together, output=[1, 2, 2, 2].

```

1920

1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953

1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994

1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036

2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123

```

Let's solve input = [3, 2, 1, -1, 2].

dp[4] = max(input[4], 0) = max(2, 0) = 2
dp[3] = max(input[3], input[4], 0) = max(-1, 2, 0) = 2
dp[2] = max(dp[3], input[2] + dp[4], 0) = max(2, 1 + 2, 0) = 3
dp[1] = max(dp[2], input[1] + dp[3], 0) = max(3, 2 + 2, 0) = 4
dp[0] = max(dp[1], input[0] + dp[2], 0) = max(4, 3 + 3, 0) = 6

Finally, we reconstruct the lexicographically smallest
subsequence that fulfills the task objective by selecting
numbers as follows. We store the result on a list named "
output".

Let can_use_next_item = True.
Since dp[0] == input[0] + dp[2] (6 == 3 + 3) and
can_use_next_item == True, we store output[0] = 1. We update
can_use_next_item = False.
Since dp[1] != input[1] + dp[3] (4 != 2 + 2) or
can_use_next_item == False, we store output[1] = 2. We update
can_use_next_item = True.
Since dp[2] == input[2] + dp[4] (3 == 1 + 2) and
can_use_next_item == True, we store output[2] = 1. We update
can_use_next_item = False.
Since dp[3] != input[3] (2 != -1) or can_use_next_item ==
False, we store output[3] = 2. We update can_use_next_item =
True.
Since dp[4] == input[4] (2 == 2) and can_use_next_item == True,
we store output[4] = 1.

Reconstructing all together, output=[1, 2, 1, 2, 1].

Let's solve input = [0, 4, -2, 3, -3, -1].

dp[5] = max(input[5], 0) = max(-1, 0) = 0
dp[4] = max(input[4], input[5], 0) = max(-3, -1, 0) = 0
dp[3] = max(dp[4], input[3] + dp[5], 0) = max(0, 3 + 0, 0) = 3
dp[2] = max(dp[3], input[2] + dp[4], 0) = max(3, -2 + 0, 0) =
3
dp[1] = max(dp[2], input[1] + dp[3], 0) = max(3, 4 + 3, 0) = 7
dp[0] = max(dp[1], input[0] + dp[2], 0) = max(7, 0 + 3, 0) = 7

Finally, we reconstruct the lexicographically smallest
subsequence that fulfills the task objective by selecting
numbers as follows. We store the result on a list named "
output".

Let can_use_next_item = True.
Since dp[0] != input[0] + dp[2] (7 != 0 + 3) or
can_use_next_item == False, we store output[0] = 2. We update
can_use_next_item = True.
Since dp[1] == input[1] + dp[3] (7 == 4 + 3) and
can_use_next_item == True, we store output[1] = 1. We update
can_use_next_item = False.
Since dp[2] != input[2] + dp[4] (3 != -2 + 0) or
can_use_next_item == False, we store output[2] = 2. We update
can_use_next_item = True.
Since dp[3] == input[3] + dp[5] (3 == 3 + 0) and
can_use_next_item == True, we store output[3] = 1. We update
can_use_next_item = False.
Since dp[4] != input[4] (0 != -3) or can_use_next_item ==
False, we store output[4] = 2. We update can_use_next_item =
True.
Since dp[5] != input[5] (0 != -1) or can_use_next_item ==
False, we store output[5] = 2.

Reconstructing all together, output=[2, 1, 2, 1, 2, 2].

{prompt}
"""

cot_implicit_6s = """Given a sequence of integers, find a
subsequence with the highest sum, such that no two numbers in
the subsequence are adjacent in the original sequence.

To indicate the selected numbers, print an array with "1" for
chosen numbers and "2" for unchosen ones. For instance, [1, 2,
2, 2, 2] implies selecting only the first number. If multiple
solutions exist, select the lexicographically smallest.

We will solve any task instance by using dynamic programming.
We define dp[i] as the maximum sum of a subsequence that does
not include adjacent elements, when considering only the

```

2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210

```

elements of the input from the i-th position onwards.

Let's solve input = [1, 1, -5, -1].

dp[3] = max(input[3], 0) = max(-1, 0) = 0
dp[2] = max(input[2], input[3], 0) = max(-5, -1, 0) = 0
dp[1] = max(dp[2], input[1] + dp[3], 0) = max(0, 1 + 0, 0) = 1
dp[0] = max(dp[1], input[0] + dp[2], 0) = max(1, 1 + 0, 0) = 1

Finally, we reconstruct the lexicographically smallest
subsequence that fulfills the task objective by selecting
numbers as follows. We store the result on a list named "
output".

Let can_use_next_item = True.
Since dp[0] == input[0] + dp[2] (1 == 1 + 0) and
can_use_next_item == True, we store output[0] = 1. We update
can_use_next_item = False.
Since dp[1] != input[1] + dp[3] (1 != 1 + 0) or
can_use_next_item == False, we store output[1] = 2. We update
can_use_next_item = True.
Since dp[2] != input[2] (0 != -5) or can_use_next_item ==
False, we store output[2] = 2. We update can_use_next_item =
True.
Since dp[3] != input[3] (0 != -1) or can_use_next_item ==
False, we store output[3] = 2.

Reconstructing all together, output=[1, 2, 2, 2].

Let's solve input = [3, 2, 1, -1, 2].

dp[4] = max(input[4], 0) = max(2, 0) = 2
dp[3] = max(input[3], input[4], 0) = max(-1, 2, 0) = 2
dp[2] = max(dp[3], input[2] + dp[4], 0) = max(2, 1 + 2, 0) = 3
dp[1] = max(dp[2], input[1] + dp[3], 0) = max(3, 2 + 2, 0) = 4
dp[0] = max(dp[1], input[0] + dp[2], 0) = max(4, 3 + 3, 0) = 6

Finally, we reconstruct the lexicographically smallest
subsequence that fulfills the task objective by selecting
numbers as follows. We store the result on a list named "
output".

Let can_use_next_item = True.
Since dp[0] == input[0] + dp[2] (6 == 3 + 3) and
can_use_next_item == True, we store output[0] = 1. We update
can_use_next_item = False.
Since dp[1] != input[1] + dp[3] (4 != 2 + 2) or
can_use_next_item == False, we store output[1] = 2. We update
can_use_next_item = True.
Since dp[2] == input[2] + dp[4] (3 == 1 + 2) and
can_use_next_item == True, we store output[2] = 1. We update
can_use_next_item = False.
Since dp[3] != input[3] (2 != -1) or can_use_next_item ==
False, we store output[3] = 2. We update can_use_next_item =
True.
Since dp[4] == input[4] (2 == 2) and can_use_next_item == True,
we store output[4] = 1.

Reconstructing all together, output=[1, 2, 1, 2, 1].

Let's solve input = [0, 4, -2, 3, -3, -1].

dp[5] = max(input[5], 0) = max(-1, 0) = 0
dp[4] = max(input[4], input[5], 0) = max(-3, -1, 0) = 0
dp[3] = max(dp[4], input[3] + dp[5], 0) = max(0, 3 + 0, 0) = 3
dp[2] = max(dp[3], input[2] + dp[4], 0) = max(3, -2 + 0, 0) =
3
dp[1] = max(dp[2], input[1] + dp[3], 0) = max(3, 4 + 3, 0) = 7
dp[0] = max(dp[1], input[0] + dp[2], 0) = max(7, 0 + 3, 0) = 7

Finally, we reconstruct the lexicographically smallest
subsequence that fulfills the task objective by selecting
numbers as follows. We store the result on a list named "
output".

Let can_use_next_item = True.
Since dp[0] != input[0] + dp[2] (7 != 0 + 3) or
can_use_next_item == False, we store output[0] = 2. We update
can_use_next_item = True.
Since dp[1] == input[1] + dp[3] (7 == 4 + 3) and
can_use_next_item == True, we store output[1] = 1. We update
can_use_next_item = False.
Since dp[2] != input[2] + dp[4] (3 != -2 + 0) or
can_use_next_item == False, we store output[2] = 2. We update

```



```

2211 can_use_next_item = True.
2212 Since dp[3] == input[3] + dp[5] (3 == 3 + 0) and
2213 can_use_next_item == True, we store output[3] = 1. We update
2214 can_use_next_item = False.
2215 Since dp[4] != input[4] (0 != -3) or can_use_next_item ==
2216 False, we store output[4] = 2. We update can_use_next_item =
2217 True.
2218 Since dp[5] != input[5] (0 != -1) or can_use_next_item ==
2219 False, we store output[5] = 2.
2220
2221 Reconstructing all together, output=[2, 1, 2, 1, 2, 2].
2222
2223
2224 Let's solve input = [-3, -4, 4, -1].
2225
2226 dp[3] = max(input[3], 0) = max(-1, 0) = 0
2227 dp[2] = max(input[2], input[3], 0) = max(4, -1, 0) = 4
2228 dp[1] = max(dp[2], input[1] + dp[3], 0) = max(4, -4 + 0, 0) =
2229 4
2230 dp[0] = max(dp[1], input[0] + dp[2], 0) = max(4, -3 + 4, 0) =
2231 4
2232
2233 Finally, we reconstruct the lexicographically smallest
2234 subsequence that fulfills the task objective by selecting
2235 numbers as follows. We store the result on a list named "
2236 output".
2237
2238 Let can_use_next_item = True.
2239 Since dp[0] != input[0] + dp[2] (4 != -3 + 4) or
2240 can_use_next_item == False, we store output[0] = 2. We update
2241 can_use_next_item = True.
2242 Since dp[1] != input[1] + dp[3] (4 != -4 + 0) or
2243 can_use_next_item == False, we store output[1] = 2. We update
2244 can_use_next_item = True.
2245 Since dp[2] == input[2] (4 == 4) and can_use_next_item == True,
2246 we store output[2] = 1. We update can_use_next_item = False.
2247 Since dp[3] != input[3] (0 != -1) or can_use_next_item ==
2248 False, we store output[3] = 2.
2249
2250 Reconstructing all together, output=[2, 2, 1, 2].
2251
2252
2253 Let's solve input = [3, 4, -3, -1, -4].
2254
2255 dp[4] = max(input[4], 0) = max(-4, 0) = 0
2256 dp[3] = max(input[3], input[4], 0) = max(-1, -4, 0) = 0
2257 dp[2] = max(dp[3], input[2] + dp[4], 0) = max(0, -3 + 0, 0) =
2258 0
2259 dp[1] = max(dp[2], input[1] + dp[3], 0) = max(0, 4 + 0, 0) = 4
2260 dp[0] = max(dp[1], input[0] + dp[2], 0) = max(4, 3 + 0, 0) = 4
2261
2262 Finally, we reconstruct the lexicographically smallest
2263 subsequence that fulfills the task objective by selecting
2264 numbers as follows. We store the result on a list named "
2265 output".
2266
2267 Let can_use_next_item = True.
2268 Since dp[0] != input[0] + dp[2] (4 != 3 + 0) or
2269 can_use_next_item == False, we store output[0] = 2. We update
2270 can_use_next_item = True.
2271 Since dp[1] == input[1] + dp[3] (4 == 4 + 0) and
2272 can_use_next_item == True, we store output[1] = 1. We update
2273 can_use_next_item = False.
2274 Since dp[2] != input[2] + dp[4] (0 != -3 + 0) or
2275 can_use_next_item == False, we store output[2] = 2. We update
2276 can_use_next_item = True.
2277 Since dp[3] != input[3] (0 != -1) or can_use_next_item ==
2278 False, we store output[3] = 2. We update can_use_next_item =
2279 True.
2280 Since dp[4] != input[4] (0 != -4) or can_use_next_item ==
2281 False, we store output[4] = 2.
2282
2283 Reconstructing all together, output=[2, 1, 2, 2, 2].
2284
2285
2286 Let's solve input = [-4, 5, 0, 2, 3, -4].
2287
2288 dp[5] = max(input[5], 0) = max(-4, 0) = 0
2289 dp[4] = max(input[4], input[5], 0) = max(3, -4, 0) = 3
2290 dp[3] = max(dp[4], input[3] + dp[5], 0) = max(3, 2 + 0, 0) = 3
2291 dp[2] = max(dp[3], input[2] + dp[4], 0) = max(3, 0 + 3, 0) = 3
2292 dp[1] = max(dp[2], input[1] + dp[3], 0) = max(3, 5 + 3, 0) = 8
2293 dp[0] = max(dp[1], input[0] + dp[2], 0) = max(8, -4 + 3, 0) =
2294 8
2295
2296 Finally, we reconstruct the lexicographically smallest
2297 subsequence that fulfills the task objective by selecting

```

```

numbers as follows. We store the result on a list named "
output".

Let can_use_next_item = True.
Since dp[0] != input[0] + dp[2] (8 != -4 + 3) or
can_use_next_item == False, we store output[0] = 2. We update
can_use_next_item = True.
Since dp[1] == input[1] + dp[3] (8 == 5 + 3) and
can_use_next_item == True, we store output[1] = 1. We update
can_use_next_item = False.
Since dp[2] != input[2] + dp[4] (3 != 0 + 3) or
can_use_next_item == False, we store output[2] = 2. We update
can_use_next_item = True.
Since dp[3] != input[3] + dp[5] (3 != 2 + 0) or
can_use_next_item == False, we store output[3] = 2. We update
can_use_next_item = True.
Since dp[4] == input[4] (3 == 3) and can_use_next_item == True,
we store output[4] = 1. We update can_use_next_item = False.
Since dp[5] != input[5] (0 != -4) or can_use_next_item ==
False, we store output[5] = 2.

Reconstructing all together, output=[2, 1, 2, 2, 1, 2].

{prompt}
"""

```

Listing 9: CoT Explicit prompts

```

cot_explicit_3s = """Given a sequence of integers, find a
subsequence with the highest sum, such that no two numbers in
the subsequence are adjacent in the original sequence.

To indicate the selected numbers, print an array with "1" for
chosen numbers and "2" for unchosen ones. For instance, [1, 2,
2, 2, 2] implies selecting only the first number. If multiple
solutions exist, select the lexicographically smallest.

We will solve any task instance by using dynamic programming.
We define dp[i] as the maximum sum of a subsequence that does
not include adjacent elements, when considering only the
elements of the input from the i-th position onwards.

Let's solve input = [1, 1, -5, -1].

There are 4 numbers in the input sequence, so we will use a
list of size 4 to store the dynamic programming values. We
initialize all values to 0.
dp[3] = max(input[3], 0) = max(-1, 0) = 0
dp[2] = max(input[2], input[3], 0) = max(-5, -1, 0) = 0
dp[1] = max(dp[2], input[1] + dp[3], 0) = max(0, 1 + 0, 0) =
max(0, 1, 0) = 1
dp[0] = max(dp[1], input[0] + dp[2], 0) = max(1, 1 + 0, 0) =
max(1, 1, 0) = 1

Finally, we reconstruct the lexicographically smallest
subsequence that fulfills the task objective by selecting
numbers as follows. We store the result on a list named "
output".

Let can_use_next_item = True.
Since dp[0]=1, input[0]=1, dp[2]=0, input[0] + dp[2] = 1 == 1
= dp[0] and can_use_next_item == True, we store output[0] = 1.
We update can_use_next_item = False.
Since can_use_next_item == False, we store output[1] = 2. We
update can_use_next_item = True.
Since dp[2] = 0, input[2] = -5, dp[2] != input[2], we store
output[2] = 2. We update can_use_next_item = True.
Since dp[3] = 0, input[3] = -1, dp[3] != input[3], we store
output[3] = 2.

Reconstructing all together, output=[1, 2, 2, 2].

Let's solve input = [3, 2, 1, -1, 2].

There are 5 numbers in the input sequence, so we will use a
list of size 5 to store the dynamic programming values. We
initialize all values to 0.
dp[4] = max(input[4], 0) = max(2, 0) = 2
dp[3] = max(input[3], input[4], 0) = max(-1, 2, 0) = 2
dp[2] = max(dp[3], input[2] + dp[4], 0) = max(2, 1 + 2, 0) =
max(2, 3, 0) = 3

```

2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323

2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380

```

2381 dp[1] = max(dp[2], input[1] + dp[3], 0) = max(3, 2 + 2, 0) =
2382 max(3, 4, 0) = 4
2383 dp[0] = max(dp[1], input[0] + dp[2], 0) = max(4, 3 + 3, 0) =
2384 max(4, 6, 0) = 6
2385
2386 Finally, we reconstruct the lexicographically smallest
2387 subsequence that fulfills the task objective by selecting
2388 numbers as follows. We store the result on a list named "
2389 output".
2390
2391 Let can_use_next_item = True.
2392 Since dp[0]=6, input[0]=3, dp[2]=3, input[0] + dp[2] = 6 == 6
2393 = dp[0] and can_use_next_item == True, we store output[0] = 1.
2394 We update can_use_next_item = False.
2395 Since can_use_next_item == False, we store output[1] = 2. We
2396 update can_use_next_item = True.
2397 Since dp[2]=3, input[2]=1, dp[4]=2, input[2] + dp[4] = 3 == 3
2398 = dp[2] and can_use_next_item == True, we store output[2] = 1.
2399 We update can_use_next_item = False.
2400 Since can_use_next_item == False, we store output[3] = 2. We
2401 update can_use_next_item = True.
2402 Since dp[4] = 2, input[4] = 2, dp[4] == input[4] and
2403 can_use_next_item == True, we store output[4] = 1.
2404
2405 Reconstructing all together, output=[1, 2, 1, 2, 1].
2406
2407
2408
2409 Let\'s solve input = [0, 4, -2, 3, -3, -1].
2410
2411 There are 6 numbers in the input sequence, so we will use a
2412 list of size 6 to store the dynamic programming values. We
2413 initialize all values to 0.
2414 dp[5] = max(input[5], 0) = max(-1, 0) = 0
2415 dp[4] = max(input[4], input[5], 0) = max(-3, -1, 0) = 0
2416 dp[3] = max(dp[4], input[3] + dp[5], 0) = max(0, 3 + 0, 0) =
2417 max(0, 3, 0) = 3
2418 dp[2] = max(dp[3], input[2] + dp[4], 0) = max(3, -2 + 0, 0) =
2419 max(3, -2, 0) = 3
2420 dp[1] = max(dp[2], input[1] + dp[3], 0) = max(3, 4 + 3, 0) =
2421 max(3, 7, 0) = 7
2422 dp[0] = max(dp[1], input[0] + dp[2], 0) = max(7, 0 + 3, 0) =
2423 max(7, 3, 0) = 7
2424
2425 Finally, we reconstruct the lexicographically smallest
2426 subsequence that fulfills the task objective by selecting
2427 numbers as follows. We store the result on a list named "
2428 output".
2429
2430 Let can_use_next_item = True.
2431 Since dp[0]=7, input[0]=0, dp[2]=3, input[0] + dp[2] = 3 != 7
2432 = dp[0], we store output[0] = 2. We update can_use_next_item =
2433 True.
2434 Since dp[1]=7, input[1]=4, dp[3]=3, input[1] + dp[3] = 7 == 7
2435 = dp[1] and can_use_next_item == True, we store output[1] = 1.
2436 We update can_use_next_item = False.
2437 Since can_use_next_item == False, we store output[2] = 2. We
2438 update can_use_next_item = True.
2439 Since dp[3]=3, input[3]=3, dp[5]=0, input[3] + dp[5] = 3 == 3
2440 = dp[3] and can_use_next_item == True, we store output[3] = 1.
2441 We update can_use_next_item = False.
2442 Since can_use_next_item == False, we store output[4] = 2. We
2443 update can_use_next_item = True.
2444 Since dp[5] = 0, input[5] = -1, dp[5] != input[5], we store
2445 output[5] = 2.
2446
2447 Reconstructing all together, output=[2, 1, 2, 1, 2, 2].
2448
2449
2450 {prompt}
2451 """
2452 cot_explicit_6s = """Given a sequence of integers, find a
2453 subsequence with the highest sum, such that no two numbers in
2454 the subsequence are adjacent in the original sequence.
2455
2456 To indicate the selected numbers, print an array with "1" for
2457 chosen numbers and "2" for unchosen ones. For instance, [1, 2,
2458 2, 2, 2] implies selecting only the first number. If multiple
2459 solutions exist, select the lexicographically smallest.
2460
2461 We will solve any task instance by using dynamic programming.
2462 We define dp[i] as the maximum sum of a subsequence that does
2463 not include adjacent elements, when considering only the
2464 elements of the input from the i-th position onwards.

```

```

Let\'s solve input = [1, 1, -5, -1].
There are 4 numbers in the input sequence, so we will use a
list of size 4 to store the dynamic programming values. We
initialize all values to 0.
dp[3] = max(input[3], 0) = max(-1, 0) = 0
dp[2] = max(input[2], input[3], 0) = max(-5, -1, 0) = 0
dp[1] = max(dp[2], input[1] + dp[3], 0) = max(0, 1 + 0, 0) =
max(0, 1, 0) = 1
dp[0] = max(dp[1], input[0] + dp[2], 0) = max(1, 1 + 0, 0) =
max(1, 1, 0) = 1
Finally, we reconstruct the lexicographically smallest
subsequence that fulfills the task objective by selecting
numbers as follows. We store the result on a list named "
output".
Let can_use_next_item = True.
Since dp[0]=1, input[0]=1, dp[2]=0, input[0] + dp[2] = 1 == 1
= dp[0] and can_use_next_item == True, we store output[0] = 1.
We update can_use_next_item = False.
Since can_use_next_item == False, we store output[1] = 2. We
update can_use_next_item = True.
Since dp[2] = 0, input[2] = -5, dp[2] != input[2], we store
output[2] = 2. We update can_use_next_item = True.
Since dp[3] = 0, input[3] = -1, dp[3] != input[3], we store
output[3] = 2.
Reconstructing all together, output=[1, 2, 2, 2].
Let\'s solve input = [3, 2, 1, -1, 2].
There are 5 numbers in the input sequence, so we will use a
list of size 5 to store the dynamic programming values. We
initialize all values to 0.
dp[4] = max(input[4], 0) = max(2, 0) = 2
dp[3] = max(input[3], input[4], 0) = max(-1, 2, 0) = 2
dp[2] = max(dp[3], input[2] + dp[4], 0) = max(2, 1 + 2, 0) =
max(2, 3, 0) = 3
dp[1] = max(dp[2], input[1] + dp[3], 0) = max(3, 2 + 2, 0) =
max(3, 4, 0) = 4
dp[0] = max(dp[1], input[0] + dp[2], 0) = max(4, 3 + 3, 0) =
max(4, 6, 0) = 6
Finally, we reconstruct the lexicographically smallest
subsequence that fulfills the task objective by selecting
numbers as follows. We store the result on a list named "
output".
Let can_use_next_item = True.
Since dp[0]=6, input[0]=3, dp[2]=3, input[0] + dp[2] = 6 == 6
= dp[0] and can_use_next_item == True, we store output[0] = 1.
We update can_use_next_item = False.
Since can_use_next_item == False, we store output[1] = 2. We
update can_use_next_item = True.
Since dp[2]=3, input[2]=1, dp[4]=2, input[2] + dp[4] = 3 == 3
= dp[2] and can_use_next_item == True, we store output[2] = 1.
We update can_use_next_item = False.
Since can_use_next_item == False, we store output[3] = 2. We
update can_use_next_item = True.
Since dp[4] = 2, input[4] = 2, dp[4] == input[4] and
can_use_next_item == True, we store output[4] = 1.
Reconstructing all together, output=[1, 2, 1, 2, 1].
Let\'s solve input = [0, 4, -2, 3, -3, -1].
There are 6 numbers in the input sequence, so we will use a
list of size 6 to store the dynamic programming values. We
initialize all values to 0.
dp[5] = max(input[5], 0) = max(-1, 0) = 0
dp[4] = max(input[4], input[5], 0) = max(-3, -1, 0) = 0
dp[3] = max(dp[4], input[3] + dp[5], 0) = max(0, 3 + 0, 0) =
max(0, 3, 0) = 3
dp[2] = max(dp[3], input[2] + dp[4], 0) = max(3, -2 + 0, 0) =
max(3, -2, 0) = 3
dp[1] = max(dp[2], input[1] + dp[3], 0) = max(3, 4 + 3, 0) =
max(3, 7, 0) = 7
dp[0] = max(dp[1], input[0] + dp[2], 0) = max(7, 0 + 3, 0) =
max(7, 3, 0) = 7
Finally, we reconstruct the lexicographically smallest
subsequence that fulfills the task objective by selecting

```

2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554

```

2555 numbers as follows. We store the result on a list named "
2556 output".
2557
2558 Let can_use_next_item = True.
2559 Since dp[0]=7, input[0]=0, dp[2]=3, input[0] + dp[2] = 3 != 7
2560 = dp[0], we store output[0] = 2. We update can_use_next_item =
2561 True.
2562 Since dp[1]=7, input[1]=4, dp[3]=3, input[1] + dp[3] = 7 == 7
2563 = dp[1] and can_use_next_item == True, we store output[1] = 1.
2564 We update can_use_next_item = False.
2565 Since can_use_next_item == False, we store output[2] = 2. We
2566 update can_use_next_item = True.
2567 Since dp[3]=3, input[3]=3, dp[5]=0, input[3] + dp[5] = 3 == 3
2568 = dp[3] and can_use_next_item == True, we store output[3] = 1.
2569 We update can_use_next_item = False.
2570 Since can_use_next_item == False, we store output[4] = 2. We
2571 update can_use_next_item = True.
2572 Since dp[5] = 0, input[5] = -1, dp[5] != input[5], we store
2573 output[5] = 2.
2574
2575 Reconstructing all together, output=[2, 1, 2, 1, 2, 2].
2576
2577
2578 Let's solve input = [-3, -4, 4, -1].
2579
2580
2581 There are 4 numbers in the input sequence, so we will use a
2582 list of size 4 to store the dynamic programming values. We
2583 initialize all values to 0.
2584 dp[3] = max(input[3], 0) = max(-1, 0) = 0
2585 dp[2] = max(input[2], input[3], 0) = max(4, -1, 0) = 4
2586 dp[1] = max(dp[2], input[1] + dp[3], 0) = max(4, -4 + 0, 0) =
2587 max(4, -4, 0) = 4
2588 dp[0] = max(dp[1], input[0] + dp[2], 0) = max(4, -3 + 4, 0) =
2589 max(4, 1, 0) = 4
2590
2591 Finally, we reconstruct the lexicographically smallest
2592 subsequence that fulfills the task objective by selecting
2593 numbers as follows. We store the result on a list named "
2594 output".
2595
2596 Let can_use_next_item = True.
2597 Since dp[0]=4, input[0]=-3, dp[2]=4, input[0] + dp[2] = 1 != 4
2598 = dp[0], we store output[0] = 2. We update can_use_next_item
2599 = True.
2600 Since dp[1]=4, input[1]=-4, dp[3]=0, input[1] + dp[3] = -4 !=
2601 4 = dp[1], we store output[1] = 2. We update can_use_next_item
2602 = True.
2603 Since dp[2] = 4, input[2] = 4, dp[2] == input[2] and
2604 can_use_next_item == True, we store output[2] = 1. We update
2605 can_use_next_item = False.
2606 Since can_use_next_item == False, we store output[3] = 2.
2607
2608 Reconstructing all together, output=[2, 2, 1, 2].
2609
2610
2611 Let's solve input = [3, 4, -3, -1, -4].
2612
2613
2614 There are 5 numbers in the input sequence, so we will use a
2615 list of size 5 to store the dynamic programming values. We
2616 initialize all values to 0.
2617 dp[4] = max(input[4], 0) = max(-4, 0) = 0
2618 dp[3] = max(input[3], input[4], 0) = max(-1, -4, 0) = 0
2619 dp[2] = max(dp[3], input[2] + dp[4], 0) = max(0, -3 + 0, 0) =
2620 max(0, -3, 0) = 0
2621 dp[1] = max(dp[2], input[1] + dp[3], 0) = max(0, 4 + 0, 0) =
2622 max(0, 4, 0) = 4
2623 dp[0] = max(dp[1], input[0] + dp[2], 0) = max(4, 3 + 0, 0) =
2624 max(4, 3, 0) = 4
2625
2626 Finally, we reconstruct the lexicographically smallest
2627 subsequence that fulfills the task objective by selecting
2628 numbers as follows. We store the result on a list named "
2629 output".
2630
2631 Let can_use_next_item = True.
2632 Since dp[0]=4, input[0]=3, dp[2]=0, input[0] + dp[2] = 3 != 4
2633 = dp[0], we store output[0] = 2. We update can_use_next_item =
2634 True.
2635 Since dp[1]=4, input[1]=4, dp[3]=0, input[1] + dp[3] = 4 == 4
2636 = dp[1] and can_use_next_item == True, we store output[1] = 1.
2637 We update can_use_next_item = False.
2638 Since can_use_next_item == False, we store output[2] = 2. We
2639 update can_use_next_item = True.
2640 Since dp[3] = 0, input[3] = -1, dp[3] != input[3], we store
2641 output[3] = 2. We update can_use_next_item = True.

```

```

Since dp[4] = 0, input[4] = -4, dp[4] != input[4], we store
output[4] = 2.
Reconstructing all together, output=[2, 1, 2, 2, 2].
Let's solve input = [-4, 5, 0, 2, 3, -4].
There are 6 numbers in the input sequence, so we will use a
list of size 6 to store the dynamic programming values. We
initialize all values to 0.
dp[5] = max(input[5], 0) = max(-4, 0) = 0
dp[4] = max(input[4], input[5], 0) = max(3, -4, 0) = 3
dp[3] = max(dp[4], input[3] + dp[5], 0) = max(3, 2 + 0, 0) =
max(3, 2, 0) = 3
dp[2] = max(dp[3], input[2] + dp[4], 0) = max(3, 0 + 3, 0) =
max(3, 3, 0) = 3
dp[1] = max(dp[2], input[1] + dp[3], 0) = max(3, 5 + 3, 0) =
max(3, 8, 0) = 8
dp[0] = max(dp[1], input[0] + dp[2], 0) = max(8, -4 + 3, 0) =
max(8, -1, 0) = 8
Finally, we reconstruct the lexicographically smallest
subsequence that fulfills the task objective by selecting
numbers as follows. We store the result on a list named "
output".
Let can_use_next_item = True.
Since dp[0]=8, input[0]=-4, dp[2]=3, input[0] + dp[2] = -1 !=
8 = dp[0], we store output[0] = 2. We update can_use_next_item
= True.
Since dp[1]=8, input[1]=5, dp[3]=3, input[1] + dp[3] = 8 == 8
= dp[1] and can_use_next_item == True, we store output[1] = 1.
We update can_use_next_item = False.
Since can_use_next_item == False, we store output[2] = 2. We
update can_use_next_item = True.
Since dp[3]=3, input[3]=2, dp[5]=0, input[3] + dp[5] = 2 != 3
= dp[3], we store output[3] = 2. We update can_use_next_item =
True.
Since dp[4] = 3, input[4] = 3, dp[4] == input[4] and
can_use_next_item == True, we store output[4] = 1. We update
can_use_next_item = False.
Since can_use_next_item == False, we store output[5] = 2.
Reconstructing all together, output=[2, 1, 2, 2, 1, 2].
{prompt}
"""

```

2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691

I.3 Comparison between CoT Implicit and CoT Explicit

2692
2693

Listing 10: Compare CoT Implicit with CoT Explicit

```

Given a sequence of integers, find a subsequence with the
highest sum, such that no two numbers in the subsequence are
adjacent in the original sequence.
To indicate the selected numbers, print an array with "1" for
chosen numbers and "2" for unchosen ones. For instance, [1, 2,
2, 2, 2] implies selecting only the first number. If multiple
solutions exist, select the lexicographically smallest.
We will solve any task instance by using dynamic programming.
We define dp[i] as the maximum sum of a subsequence that does
not include adjacent elements, when considering only the
elements of the input from the i-th position onwards.
Let's solve input = [-45, 40, 77, -93, 25, -73].
###
Explicit: There are 6 numbers in the input sequence, so we
will use a list of size 6 to store the dynamic programming
values. We initialize all values to 0.
dp[5] = max(input[5], 0) = max(-73, 0) = 0
dp[4] = max(input[4], input[5], 0) = max(25, -73, 0) = 25
Implicit: dp[3] = max(dp[4], input[3] + dp[5], 0) = max(25,
-93 + 0, 0) = 25
Explicit: dp[3] = max(dp[4], input[3] + dp[5], 0) = max(25,
-93 + 0, 0) = max(25, -93, 0) = 25
Implicit: dp[2] = max(dp[3], input[2] + dp[4], 0) = max(25, 77
+ 25, 0) = 102

```

2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722

2723 Explicit: $dp[2] = \max(dp[3], input[2] + dp[4], 0) = \max(25, 77$
 2724 $+ 25, 0) = \max(25, 102, 0) = 102$
 2725 Implicit: $dp[1] = \max(dp[2], input[1] + dp[3], 0) = \max(102,$
 2726 $40 + 25, 0) = 102$
 2727 Explicit: $dp[1] = \max(dp[2], input[1] + dp[3], 0) = \max(102,$
 2728 $40 + 25, 0) = \max(102, 65, 0) = 102$
 2729 Implicit: $dp[0] = \max(dp[1], input[0] + dp[2], 0) = \max(102,$
 2730 $-45 + 102, 0) = 102$
 2731 Explicit: $dp[0] = \max(dp[1], input[0] + dp[2], 0) = \max(102,$
 2732 $-45 + 102, 0) = \max(102, 57, 0) = 102$
 2733
 2734 Finally, we reconstruct the lexicographically smallest
 2735 subsequence that fulfills the task objective by selecting
 2736 numbers as follows. We store the result on a list named "output".
 2737
 2738 Let `can_use_next_item = True`.
 2739 Implicit: Since $dp[0] \neq input[0] + dp[2]$ ($102 \neq -45 + 102$)
 2740 or `can_use_next_item == False`, we store `output[0] = 2`. We
 2741 update `can_use_next_item = True`.
 2742 Explicit: Since $dp[0] = 102$, $input[0] = -45$, $dp[2] = 102$, $input[0] +$
 2743 $dp[2] = 57 \neq 102 = dp[0]$, we store `output[0] = 2`. We update
 2744 `can_use_next_item = True`.
 2745 Implicit: Since $dp[1] \neq input[1] + dp[3]$ ($102 \neq 40 + 25$) or
 2746 `can_use_next_item == False`, we store `output[1] = 2`. We update
 2747 `can_use_next_item = True`.
 2748 Explicit: Since $dp[1] = 102$, $input[1] = 40$, $dp[3] = 25$, $input[1] +$
 2749 $dp[3] = 65 \neq 102 = dp[1]$, we store `output[1] = 2`. We update
 2750 `can_use_next_item = True`.
 2751 Implicit: Since $dp[2] \neq input[2] + dp[4]$ ($102 \neq 77 + 25$) and
 2752 `can_use_next_item == True`, we store `output[2] = 1`. We update
 2753 `can_use_next_item = False`.
 2754 Explicit: Since $dp[2] = 102$, $input[2] = 77$, $dp[4] = 25$, $input[2] +$
 2755 $dp[4] = 102 \neq 102 = dp[2]$ and `can_use_next_item == True`, we
 2756 store `output[2] = 1`. We update `can_use_next_item = False`.
 2757 Implicit: Since $dp[3] \neq input[3] + dp[5]$ ($25 \neq -93 + 0$) or
 2758 `can_use_next_item == False`, we store `output[3] = 2`. We update
 2759 `can_use_next_item = True`.
 2760 Explicit: Since `can_use_next_item == False`, we store `output[3]`
 2761 `= 2`. We update `can_use_next_item = True`.
 2762 Implicit: Since $dp[4] \neq input[4]$ ($25 \neq 25$) and
 2763 `can_use_next_item == True`, we store `output[4] = 1`. We update
 2764 `can_use_next_item = False`.
 2765 Explicit: Since $dp[4] = 25$, $input[4] = 25$, $dp[4] \neq input[4]$
 2766 and `can_use_next_item == True`, we store `output[4] = 1`. We
 2767 update `can_use_next_item = False`.
 2768 Implicit: Since $dp[5] \neq input[5]$ ($0 \neq -73$) or
 2769 `can_use_next_item == False`, we store `output[5] = 2`.
 2770 Explicit: Since `can_use_next_item == False`, we store `output[5]`
 2771 `= 2`.
 2772 Reconstructing all together, `output=[2, 2, 1, 2, 1, 2]`.

2775 I.4 Travel planning prompts

Listing 11: CoT prompts

2776 `prompt_cot_zero_shot = ""`
 2777 The user will ask for a flight route between two cities. You
 2778 need to generate a response with the route. Your response
 2779 should be in the format "[city 1]-[city 2]-[city 3]-...-[city
 2780 n]". If there is no solution, reply "Answer: None."
 2781 Question: {input}
 2782 Answer: ""
 2783
 2784 `prompt_cot_1s = ""`
 2785 The user will ask for a flight route between two cities. You
 2786 need to generate a response with the route. Your response
 2787 should be in the format "Answer: [city 1]-[city 2]-[city
 2788 3]-...-[city n]". If there is no solution, reply "Answer: None."
 2789
 2790 Question: What is the flight route from Dublin to Sydney?
 2791 Answer: Dublin-London-Sydney.
 2792 Question: {input}
 2793 Answer: ""
 2794
 2795 `prompt_cot_3s = ""`
 2796 The user will ask for a flight route between two cities. You
 2797 need to generate a response with the route. Your response
 2798 should be in the format "[city 1]-[city 2]-[city 3]-...-[city
 2799 n]". If there is no solution, reply "Answer: None."
 2800 Question: What is the flight route from Dublin to Sydney?
 2801 Answer: Dublin-London-Sydney.
 2802 Question: What is the flight route from New York to Amsterdam?
 2803 Answer: New York-London-Amsterdam.

2804 Question: What is the flight route from Toronto to Sydney?
 2805 Answer: Toronto-San Francisco-Sydney.
 2806 Question: {input}
 2807 Answer: ""
 2808
 2809 `prompt_cot_8s = ""`
 2810 The user will ask for a flight route between two cities. You
 2811 need to generate a response with the route. Your response
 2812 should be in the format "[city 1]-[city 2]-[city 3]-...-[city
 2813 n]". If there is no solution, reply "Answer: None."
 2814 Question: What is the flight route from Dublin to Sydney?
 2815 Answer: Dublin-London-Sydney.
 2816 Question: What is the flight route from New York to Amsterdam?
 2817 Answer: New York-London-Amsterdam.
 2818 Question: What is the flight route from Toronto to Sydney?
 2819 Answer: Toronto-San Francisco-Sydney.
 2820 Question: What is the flight route from Astana to Rome?
 2821 Answer: Astana-Moscow-Rome.
 2822 Question: What is the flight route from Visakhapatnam to
 2823 Odense?
 2824 Answer: Visakhapatnam-Hyderabad-Copenhagen-Odense.
 2825 Question: What is the flight route from Shanghai to Nanjing?
 2826 Answer: Shanghai-Nanjing.
 2827 Question: What is the flight route from Singapore to Taipei?
 2828 Answer: Singapore-Taipei.
 2829 Question: What is the flight route from Sydney to Istanbul?
 2830 Answer: Sydney-Singapore-Istanbul.
 2831 Question: {input}
 2832 Answer: ""

Listing 12: ToT prompts

2833 `prompt_tot_propose_zero_shot = ""` List a few possible cities
 2834 to fly to from the current city via one direct flight. If the
 2835 goal city can be reached via one direct flight from the
 2836 current city, just answer the goal city. Format of your
 2837 response is "Answer: [city 1], [city 2], [city 3], ... [city n
 2838]."
 2839 Question: {input}
 2840 ""
 2841
 2842 `prompt_tot_propose_1s = ""` List the a few possible cities to
 2843 fly to from the current city via one direct flight. If the
 2844 goal city can be reached via one direct flight from the
 2845 current city, just answer the goal city. Format of your
 2846 response is "Answer: [city 1], [city 2], [city 3], ... [city n
 2847]."
 2848 Question: You want to go to Sydney and you are at Dublin.
 2849 Propose a few possible cities with direct flights to go to for
 2850 the next step.
 2851 Answer: London, Paris, Frankfurt, Amsterdam, Zurich.
 2852 Question: {input}
 2853 ""
 2854
 2855 `prompt_tot_propose_3s = ""` List the a few possible cities to
 2856 fly to from the current city via one direct flight. If the
 2857 goal city can be reached via one direct flight from the
 2858 current city, just answer the goal city. Format of your
 2859 response is "Answer: [city 1], [city 2], [city 3], ... [city n
 2860]."
 2861 Question: You want to go to Sydney and you are at Dublin.
 2862 Propose a few possible cities with direct flights to go to for
 2863 the next step.
 2864 Answer: London, Paris, Mumbai.
 2865 Question: You want to go to Nanjing and you are at Shanghai.
 2866 Propose a few possible cities with direct flights to go to for
 2867 the next step.
 2868 Answer: Nanjing.
 2869 Question: You want to go to Amsterdam and you are at New York.
 2870 Propose a few possible cities with direct flights to go to
 2871 for the next step.
 2872 Answer: London, Paris, Frankfurt, Amsterdam.
 2873 Question: {input}
 2874 ""
 2875
 2876 `prompt_tot_propose_8s = ""` List the a few possible cities to
 2877 fly to from the current city via one direct flight. If the
 2878 goal city can be reached via one direct flight from the
 2879 current city, just answer the goal city. Format of your
 2880 response is "Answer: [city 1], [city 2], [city 3], ... [city n
 2881]."
 2882 Question: You want to go to Sydney and you are at Dublin.
 2883 Propose a few possible cities with direct flights to go to for
 2884 the next step.
 2885 Answer: London, Paris, Mumbai.
 2886 Question: You want to go to Amsterdam and you are at New York.

2887 Propose a few possible cities with direct flights to go to
 2888 for the next step.
 2889 Answer: London, Paris, Frankfurt.
 2890 Question: You want to go to Sydney and you are at Toronto.
 2891 Propose a few possible cities with direct flights to go to for
 2892 the next step.
 2893 Answer: San Francisco, Los Angeles, Vancouver.
 2894 Question: You want to go to Nanjing and you are at Shanghai.
 2895 Propose a few possible cities with direct flights to go to for
 2896 the next step.
 2897 Answer: Nanjing.
 2898 Question: You want to go to Rome and you are at Astana.
 2899 Propose a few possible cities with direct flights to go to for
 2900 the next step.
 2901 Answer: Moscow, Rome, Istanbul.
 2902 Question: You want to go to Odense and you are at
 2903 Visakhapatnam. Propose a few possible cities with direct
 2904 flights to go to for the next step.
 2905 Answer: Hyderabad, Copenhagen, Odense.
 2906 Question: You want to go to Taipei and you are at Singapore.
 2907 Propose a few possible cities with direct flights to go to for
 2908 the next step.
 2909 Answer: Taipei.
 2910 Question: You want to go to Istanbul and you are at Sydney.
 2911 Propose a few possible cities with direct flights to go to for
 2912 the next step.
 2913 Answer: Singapore, Dubai, Abu Dhabi.
 2914 Question: {input}
 2915 ''

Listing 13: ToT Linear prompts

2916 prompt_tot_linear_zero_shot = ""The user will ask for a
 2917 flight route between two cities. You need to generate a
 2918 response with the route.
 2919 You are simulating bfs process to find the route between two
 2920 cities. In the beginning, you have a queue ['start city'] and
 2921 an empty explored list []. You need to proceed with the
 2922 following steps:
 2923 1. Take the first city in the queue as the current city. If
 2924 the city is in the explored list, skip it. Otherwise, put the
 2925 city into the explored list.
 2926 2. Propose the possible cities with direct flights to go to
 2927 for the next step. Do not propose the explored cities and
 2928 cities in the queue.
 2929 3. Put the cities into the queue.
 2930 Repeat steps 1-3 until the goal city is included in the queue.
 2931 Respond with reasoning steps, and end with the answer, in the
 2932 format "Answer: [city 1]-[city 2]-[city 3]-...-[city n]"
 2933 Question: {input}
 2934 Let's think step by step.
 2935 ""
 2936
 2937
 2938 prompt_tot_linear_cot_1s = ""The user will ask for a flight
 2939 route between two cities. You need to generate a response with
 2940 the route.
 2941 You are simulating bfs process to find the route between two
 2942 cities. In the beginning, you have a queue ['start city'], and
 2943 you need to proceed the following steps:
 2944 1. Take the first city in the queue as the current city.
 2945 2. Propose the possible cities with direct flights to go to
 2946 for the next step. Do not propose the explored cities and
 2947 cities in the queue.
 2948 3. Put the cities into the queue.
 2949 Repeat steps 1-3 until the goal city is included in the queue.
 2950 Respond with reasoning steps, and end with the answer, in the
 2951 format "Answer: [city 1]-[city 2]-[city 3]-...-[city n]"
 2952 Question: What is the flight route from Guatemala City to
 2953 Guangzhou?
 2954 The queue is [Guatemala City]. Take the first path, Guatemala
 2955 City, from the queue.
 2956 The current city is Guatemala City, which is not in the
 2957 explored list. Thus, put the current city into the explored
 2958 list. The explored list is [Guatemala City]
 2959 The current city is Guatemala City and the goal is Guangzhou.
 2960 For the next step, the promising cities to go to are [New York,
 2961 Los Angeles, Mexico City].
 2962 Putting those cities into the queue. The queue is [Guatemala
 2963 City-New York, Guatemala City-Los Angeles, Guatemala City-
 2964 Mexico City].
 2965 Take the first path, Guatemala City-New York, from the queue.
 2966 The current city is New York, which is not in the explored
 2967 list. Thus, put the current city into the explored list. The
 2968 explored list is [Guatemala City, New York]
 2969 The current city is New York and the goal is Guangzhou. For

the next step, the promising cities to go to are [Helsinki,
 Guangzhou, Lahore].
 The goal city is Guangzhou. Since Guangzhou is in the found,
 and the current selected path is Guatemala City-New York, the
 route is Guatemala City-New York-Guangzhou.
 Answer: Guatemala City-New York-Guangzhou
 Question: {input}
 Let's think step by step.
 ""
 prompt_tot_linear_cot_2s = ""The user will ask for a flight
 route between two cities. You need to generate a response with
 the route.
 You are simulating bfs process to find the route between two
 cities. In the beginning, you have a queue ['start city'], and
 you need to proceed the following steps:
 1. Take the first city in the queue as the current city.
 2. Propose the possible cities with direct flights to go to
 for the next step. Do not propose the explored cities and
 cities in the queue.
 3. Put the cities into the queue.
 Repeat steps 1-3 until the goal city is included in the queue.
 Respond with reasoning steps, and end with the answer, in the
 format "Answer: [city 1]-[city 2]-[city 3]-...-[city n]"
 Question: What is the flight route from Guatemala City to
 Guangzhou?
 The queue is [Guatemala City]. Take the first path, Guatemala
 City, from the queue.
 The current city is Guatemala City, which is not in the
 explored list. Thus, put the current city into the explored
 list. The explored list is [Guatemala City]
 The current city is Guatemala City and the goal is Guangzhou.
 For the next step, the promising cities to go to are [New York,
 Los Angeles, Mexico City].
 Putting those cities into the queue. The queue is [Guatemala
 City-New York, Guatemala City-Los Angeles, Guatemala City-
 Mexico City].
 Take the first path, Guatemala City-New York, from the queue.
 The current city is New York, which is not in the explored
 list. Thus, put the current city into the explored list. The
 explored list is [Guatemala City, New York]
 The current city is New York and the goal is Guangzhou. For
 the next step, the promising cities to go to are [Helsinki,
 Guangzhou, Lahore].
 The goal city is Guangzhou. Since Guangzhou is in the found,
 and the current selected path is Guatemala City-New York, the
 route is Guatemala City-New York-Guangzhou.
 Answer: Guatemala City-New York-Guangzhou
 Question: What is the flight route from Tegucigalpa to
 Helsinki?
 The queue is [Tegucigalpa]. Take the first path, Tegucigalpa,
 from the queue.
 The current city is Tegucigalpa, which is not in the explored
 list. Thus, put the current city into the explored list. The
 explored list is [Tegucigalpa]
 The current city is Tegucigalpa and the goal is Helsinki. For
 the next step, the promising cities to go to are [Guatemala
 City, Miami].
 Putting those cities into the queue. The queue is [Tegucigalpa-
 Guatemala City, Tegucigalpa-Miami].
 Take the first path, Tegucigalpa-Guatemala City, from the
 queue.
 The current city is Guatemala City, which is not in the
 explored list. Thus, put the current city into the explored
 list. The explored list is [Tegucigalpa, Guatemala City]
 The current city is Guatemala City and the goal is Helsinki.
 For the next step, the promising cities to go to are [New York,
 Los Angeles, Mexico City].
 Putting those cities into the queue. The queue is [Tegucigalpa-
 Miami, Tegucigalpa-Guatemala City-New York, Tegucigalpa-
 Guatemala City-Los Angeles, Tegucigalpa-Guatemala City-
 Mexico City].
 Take the first path, Tegucigalpa-Miami, from the queue.
 The current city is Miami, which is not in the explored list.
 Thus, put the current city into the explored list. The
 explored list is [Tegucigalpa, Guatemala City, Miami]
 The current city is Miami and the goal is Helsinki. For the
 next step, the promising cities to go to are [Sao Paulo,
 Buenos Aires, Chicago].
 Putting those cities into the queue. The queue is [Tegucigalpa-
 Guatemala City-New York, Tegucigalpa-Guatemala City-Los
 Angeles, Tegucigalpa-Guatemala City-Mexico City, Tegucigalpa-
 Miami-Sao Paulo, Tegucigalpa-Miami-Buenos Aires, Tegucigalpa-
 Miami-Chicago].
 Take the first path, Tegucigalpa-Guatemala City-New York, from
 the queue.
 The current city is New York, which is not in the explored

2970
 2971
 2972
 2973
 2974
 2975
 2976
 2977
 2978
 2979
 2980
 2981
 2982
 2983
 2984
 2985
 2986
 2987
 2988
 2989
 2990
 2991
 2992
 2993
 2994
 2995
 2996
 2997
 2998
 2999
 3000
 3001
 3002
 3003
 3004
 3005
 3006
 3007
 3008
 3009
 3010
 3011
 3012
 3013
 3014
 3015
 3016
 3017
 3018
 3019
 3020
 3021
 3022
 3023
 3024
 3025
 3026
 3027
 3028
 3029
 3030
 3031
 3032
 3033
 3034
 3035
 3036
 3037
 3038
 3039
 3040
 3041
 3042
 3043
 3044
 3045
 3046
 3047
 3048
 3049
 3050
 3051
 3052
 3053
 3054
 3055
 3056

```

3057 list. Thus, put the current city into the explored list. The
3058 explored list is [Tegucigalpa, Guatemala City, Miami, New York
3059 ]
3060 The current city is New York and the goal is Helsinki. For the
3061 next step, the promising cities to go to are [Helsinki,
3062 Guangzhou, Lahore].
3063 The goal city is Helsinki. Since Helsinki is in the found, and
3064 the current selected path is Tegucigalpa-Guatemala City-New
3065 York, the route is Tegucigalpa-Guatemala City-New York-
3066 Helsinki.
3067 Answer: Tegucigalpa-Guatemala City-New York-Helsinki
3068 Question: {input}
3069 Let's think step by step.
3070 """

```

```

Input: 1 4 8 8
Steps:
8 / 4 = 2 (left: 1 2 8)
1 + 2 = 3 (left: 3 8)
3 * 8 = 24 (left: 24)
Answer: (1 + 8 / 4) * 8 = 24
Input: 5 5 5 9
Steps:
5 + 5 = 10 (left: 5 9 10)
10 + 5 = 15 (left: 9 15)
15 + 9 = 24 (left: 24)
Answer: ((5 + 5) + 5) + 9 = 24
Input: {input}
"""

```

```

3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151

```

3071

I.5 Game of 24 prompts

Listing 14: CoT prompts

```

3072 cot_prompt_1s = '''Use numbers and basic arithmetic operations
3073 (+ - * /) to obtain 24. Each step, you are only allowed to
3074 choose two of the remaining numbers to obtain a new number.
3075 Please strictly follow the format of the example. Do not
3076 include unnecessary information in your output. Do not include
3077 serial numbers that are not in the example.
3078 Input: 4 4 6 8
3079 Steps:
3080 4 + 8 = 12 (left: 4 6 12)
3081 6 - 4 = 2 (left: 2 12)
3082 2 * 12 = 24 (left: 24)
3083 Answer: (6 - 4) * (4 + 8) = 24
3084 Input: {input}
3085 '''
3086
3087 cot_prompt_3s = '''Use numbers and basic arithmetic operations
3088 (+ - * /) to obtain 24. Each step, you are only allowed to
3089 choose two of the remaining numbers to obtain a new number.
3090 Please strictly follow the format of the example. Do not
3091 include unnecessary information in your output. Do not include
3092 serial numbers that are not in the example.
3093 Input: 4 4 6 8
3094 Steps:
3095 4 + 8 = 12 (left: 4 6 12)
3096 6 - 4 = 2 (left: 2 12)
3097 2 * 12 = 24 (left: 24)
3098 Answer: (6 - 4) * (4 + 8) = 24
3099 Input: 2 9 10 12
3100 Steps:
3101 12 * 2 = 24 (left: 9 10 24)
3102 10 - 9 = 1 (left: 1 24)
3103 24 * 1 = 24 (left: 24)
3104 Answer: (12 * 2) * (10 - 9) = 24
3105 Input: 4 9 10 13
3106 Steps:
3107 13 - 10 = 3 (left: 3 4 9)
3108 9 - 3 = 6 (left: 4 6)
3109 4 * 6 = 24 (left: 24)
3110 Answer: 4 * (9 - (13 - 10)) = 24
3111 Input: {input}
3112 '''
3113
3114 cot_prompt_5s = '''Use numbers and basic arithmetic operations
3115 (+ - * /) to obtain 24. Each step, you are only allowed to
3116 choose two of the remaining numbers to obtain a new number.
3117 Please strictly follow the format of the example. Do not
3118 include unnecessary information in your output. Do not include
3119 serial numbers that are not in the example.
3120 Input: 4 4 6 8
3121 Steps:
3122 4 + 8 = 12 (left: 4 6 12)
3123 6 - 4 = 2 (left: 2 12)
3124 2 * 12 = 24 (left: 24)
3125 Answer: (6 - 4) * (4 + 8) = 24
3126 Input: 2 9 10 12
3127 Steps:
3128 12 * 2 = 24 (left: 9 10 24)
3129 10 - 9 = 1 (left: 1 24)
3130 24 * 1 = 24 (left: 24)
3131 Answer: (12 * 2) * (10 - 9) = 24
3132 Input: 4 9 10 13
3133 Steps:
3134 13 - 10 = 3 (left: 3 4 9)
3135 9 - 3 = 6 (left: 4 6)
3136 4 * 6 = 24 (left: 24)
3137 Answer: 4 * (9 - (13 - 10)) = 24

```

Listing 15: ToT prompts

```

propose_prompt_1s = '''Use numbers and basic arithmetic
operations (+ - * /) to propose possible next steps of
operation. Each step, you are only allowed to choose two of
the input numbers to obtain a new number.
Do not include serial numbers that are not in the example. Do
not include unnecessary information in your output.
Input: 2 8 8 14
Possible next steps:
2 + 8 = 10 (left: 8 10 14)
8 / 2 = 4 (left: 4 8 14)
14 + 2 = 16 (left: 8 8 16)
2 * 8 = 16 (left: 8 14 16)
8 - 2 = 6 (left: 6 8 14)
14 - 8 = 6 (left: 2 6 8)
14 / 2 = 7 (left: 7 8 8)
14 - 2 = 12 (left: 8 8 12)
Input: {input}
Possible next steps:
'''
propose_prompt_3s = '''Use numbers and basic arithmetic
operations (+ - * /) to propose possible next steps of
operation. Each step, you are only allowed to choose two of
the input numbers to obtain a new number.
Do not include serial numbers that are not in the example. Do
not include unnecessary information in your output.
Input: 2 8 8 14
Possible next steps:
2 + 8 = 10 (left: 8 10 14)
8 / 2 = 4 (left: 4 8 14)
14 + 2 = 16 (left: 8 8 16)
2 * 8 = 16 (left: 8 14 16)
8 - 2 = 6 (left: 6 8 14)
14 - 8 = 6 (left: 2 6 8)
14 / 2 = 7 (left: 7 8 8)
14 - 2 = 12 (left: 8 8 12)
Input: 1 2 7 10
Possible next steps:
1 + 2 = 3 (left: 3 7 10)
2 + 7 = 9 (left: 1 9 10)
7 + 10 = 17 (left: 1 2 17)
1 * 2 = 2 (left: 2 7 10)
2 * 7 = 14 (left: 1 14 10)
7 * 10 = 70 (left: 1 2 70)
1 - 2 = -1 (left: -1 7 10)
2 - 7 = -5 (left: 1 -5 10)
7 - 10 = -3 (left: 1 2 -3)
1 / 2 = 0.5 (left: 0.5 7 10)
2 / 7 = 0.29 (left: 1 0.29 10)
7 / 10 = 0.7 (left: 1 2 0.7)
Input: 4 4 6 8
Possible next steps:
4 + 4 = 8 (left: 6 8 8)
4 + 6 = 10 (left: 8 10 8)
6 + 8 = 14 (left: 4 14 8)
4 * 4 = 16 (left: 6 8 16)
4 * 6 = 24 (left: 8 24 8)
6 * 8 = 48 (left: 4 48 8)
4 - 4 = 0 (left: 0 6 8)
4 - 6 = -2 (left: -2 8 8)
6 - 8 = -2 (left: 4 -2 8)
4 / 4 = 1 (left: 1 6 8)
4 / 6 = 0.67 (left: 8 0.67 8)
6 / 8 = 0.75 (left: 4 0.75 8)
Input: {input}
Possible next steps:
'''
propose_prompt_5s = '''Use numbers and basic arithmetic
operations (+ - * /) to propose possible next steps of

```

```

3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221

```

3222 operation. Each step, you are only allowed to choose two of
 3223 the input numbers to obtain a new number.
 3224 Do not include serial numbers that are not in the example. Do
 3225 not include unnecessary information in your output.
 3226 Input: 2 8 8 14
 3227 Possible next steps:
 3228 $2 + 8 = 10$ (left: 8 10 14)
 3229 $8 / 2 = 4$ (left: 4 8 14)
 3230 $14 + 2 = 16$ (left: 8 8 16)
 3231 $2 * 8 = 16$ (left: 8 14 16)
 3232 $8 - 2 = 6$ (left: 6 8 14)
 3233 $14 - 8 = 6$ (left: 2 6 8)
 3234 $14 / 2 = 7$ (left: 7 8 8)
 3235 $14 - 2 = 12$ (left: 8 8 12)
 3236 Input: 1 2 7 10
 3237 Possible next steps:
 3238 $1 + 2 = 3$ (left: 3 7 10)
 3239 $2 + 7 = 9$ (left: 1 9 10)
 3240 $7 + 10 = 17$ (left: 1 2 17)
 3241 $1 * 2 = 2$ (left: 2 7 10)
 3242 $2 * 7 = 14$ (left: 1 14 10)
 3243 $7 * 10 = 70$ (left: 1 2 70)
 3244 $1 - 2 = -1$ (left: -1 7 10)
 3245 $2 - 7 = -5$ (left: 1 -5 10)
 3246 $7 - 10 = -3$ (left: 1 2 -3)
 3247 $1 / 2 = 0.5$ (left: 0.5 7 10)
 3248 $2 / 7 = 0.29$ (left: 1 0.29 10)
 3249 $7 / 10 = 0.7$ (left: 1 2 0.7)
 3250 Input: 4 4 6 8
 3251 Possible next steps:
 3252 $4 + 4 = 8$ (left: 6 8 8)
 3253 $4 + 6 = 10$ (left: 8 10 8)
 3254 $6 + 8 = 14$ (left: 4 14 8)
 3255 $4 * 4 = 16$ (left: 6 8 16)
 3256 $4 * 6 = 24$ (left: 8 24 8)
 3257 $6 * 8 = 48$ (left: 4 48 8)
 3258 $4 - 4 = 0$ (left: 0 6 8)
 3259 $4 - 6 = -2$ (left: -2 8 8)
 3260 $6 - 8 = -2$ (left: 4 -2 8)
 3261 $4 / 4 = 1$ (left: 1 6 8)
 3262 $4 / 6 = 0.67$ (left: 8 0.67 8)
 3263 $6 / 8 = 0.75$ (left: 4 0.75 8)
 3264 Input: 3 4 5 6
 3265 Possible next steps:
 3266 $3 + 4 = 7$ (left: 5 6 7)
 3267 $4 + 5 = 9$ (left: 6 9 7)
 3268 $4 + 6 = 10$ (left: 5 10 7)
 3269 $5 + 6 = 11$ (left: 4 11 7)
 3270 $3 * 4 = 12$ (left: 5 6 12)
 3271 $4 * 5 = 20$ (left: 6 20 7)
 3272 $4 * 6 = 24$ (left: 5 24 7)
 3273 $5 * 6 = 30$ (left: 4 30 7)
 3274 $3 - 4 = -1$ (left: -1 5 6)
 3275 $4 - 5 = -1$ (left: 6 -1 7)
 3276 $4 - 6 = -2$ (left: 5 -2 7)
 3277 $5 - 6 = -1$ (left: 4 -1 7)
 3278 $3 / 4 = 0.75$ (left: 0.75 5 6)
 3279 $4 / 5 = 0.8$ (left: 6 0.8 7)
 3280 $4 / 6 = 0.67$ (left: 5 0.67 7)
 3281 $5 / 6 = 0.83$ (left: 4 0.83 7)
 3282 Input: 2 4 6
 3283 Possible next steps:
 3284 $2 + 4 = 6$ (left: 6 6)
 3285 $4 + 6 = 10$ (left: 6 10)
 3286 $2 * 4 = 8$ (left: 6 8)
 3287 $4 * 6 = 24$ (left: 6 24)
 3288 $2 - 4 = -2$ (left: -2 6)
 3289 $4 - 6 = -2$ (left: 8 -2)
 3290 $2 / 4 = 0.5$ (left: 0.5 6)
 3291 $4 / 6 = 0.67$ (left: 8 0.67)
 3292 Input: {input}
 3293 Possible next steps:
 3294 '''

Output:
 (1) (2) 7 10
 1 (2) (7) 10
 1 2 (7) (10)
 (1) 2 (7) 10
 (1) 2 7 (10)
 1 (2) 7 (10)
 Input: {input}
 Select all combinations of two numbers using bracket.
 Output:
 '''
 select_prompt_3s = '''Select two numbers using the bracket.
 For example, (2) 8 8 (14) means select 2 and 14. Follow the
 format of the example.
 Do not include serial numbers that are not in the example. Do
 not include unnecessary information in your output.
 Input: 7 8 9
 Select all combinations of two numbers using bracket.
 Output:
 (7) (8) 9
 7 (8) (9)
 (7) 8 (9)
 Input: 2.33 6
 Output:
 (2.33) (6)
 Input: 1 2 7 10
 Select all combinations of two numbers using bracket.
 Output:
 (1) (2) 7 10
 1 (2) (7) 10
 1 2 (7) (10)
 (1) 2 (7) 10
 (1) 2 7 (10)
 1 (2) 7 (10)
 Input: {input}
 Select all combinations of two numbers using bracket.
 Output:
 '''
 select_prompt_5s = '''Select two numbers using the bracket.
 For example, (2) 8 8 (14) means select 2 and 14. Follow the
 format of the example.
 Do not include serial numbers that are not in the example. Do
 not include unnecessary information in your output.
 Input: 7 8 9
 Select all combinations of two numbers using bracket.
 Output:
 (7) (8) 9
 7 (8) (9)
 (7) 8 (9)
 Input: 2.33 6
 Output:
 (2.33) (6)
 Input: 1 2 7 10
 Select all combinations of two numbers using bracket.
 Output:
 (1) (2) 7 10
 1 (2) (7) 10
 1 2 (7) (10)
 (1) 2 (7) 10
 (1) 2 7 (10)
 1 (2) 7 (10)
 Input: 0.66 8 9
 Select all combinations of two numbers using bracket.
 Output:
 (0.66) (8) 9
 0.66 (8) (9)
 (0.66) 8 (9)
 Input: 2 8 8 14
 Select all combinations of two numbers using bracket.
 Output:
 (2) (8) 8 14
 2 (8) (8) 14
 2 8 (8) (14)
 (2) 8 (8) 14
 (2) 8 8 (14)
 2 (8) 8 (14)
 Input: {input}
 Select all combinations of two numbers using bracket.
 Output:
 '''
 propose_prompt_1s = '''Use the two numbers in the bracket and
 basic arithmetic operations to propose possible next steps.

3303
 3304
 3305
 3306
 3307
 3308
 3309
 3310
 3311
 3312
 3313
 3314
 3315
 3316
 3317
 3318
 3319
 3320
 3321
 3322
 3323
 3324
 3325
 3326
 3327
 3328
 3329
 3330
 3331
 3332
 3333
 3334
 3335
 3336
 3337
 3338
 3339
 3340
 3341
 3342
 3343
 3344
 3345
 3346
 3347
 3348
 3349
 3350
 3351
 3352
 3353
 3354
 3355
 3356
 3357
 3358
 3359
 3360
 3361
 3362
 3363
 3364
 3365
 3366
 3367
 3368
 3369
 3370
 3371
 3372
 3373
 3374
 3375
 3376
 3377
 3378
 3379
 3380
 3381
 3382
 3383
 3384
 3385
 3386
 3387
 3388
 3389

3295 I.5.1 ToT Decomp prompts

Listing 16: ToT Decomp prompts

3296 select_prompt_1s = '''Select two numbers using the bracket.
 3297 For example, (2) 8 8 (14) means select 2 and 14. Follow the
 3298 format of the example.
 3299 Do not include serial numbers that are not in the example. Do
 3300 not include unnecessary information in your output.
 3301 Input: 1 2 7 10
 3302 Select all combinations of two numbers using bracket.

3390 Then, remove the selected numbers by the new number. Use the
3391 format '(left: ...)' to present the remaining numbers.
3392 Do not include serial numbers that are not in the example. Do
3393 not include unnecessary information in your output.
3394 Input: (2) 8 8 (14)
3395 2 + 14 = 16, replace 2 14 by 16 (left: 8 8 16)
3396 2 * 14 = 28, replace 2 14 by 28 (left: 8 8 28)
3397 2 / 14 = 0.14, replace 2 14 by 0.14 (left: 8 8 0.14)
3398 14 / 2 = 7, replace 2 14 by 7 (left: 8 8 7)
3399 14 - 2 = 12, replace 2 14 by 12 (left: 8 8 12)
3400 2 - 14 = -12, replace 2 14 by -12 (left: 8 8 -12)
3401 Input: {input}
3402 '''
3403
3404 propose_prompt_3s = '''Use the two numbers in the bracket and
3405 basic arithmetic operations to propose possible next steps.
3406 Then, remove the selected numbers by the new number. Use the
3407 format '(left: ...)' to present the remaining numbers.
3408 Do not include serial numbers that are not in the example. Do
3409 not include unnecessary information in your output.
3410 Input: (2) 8 8 (14)
3411 2 + 14 = 16, replace 2 14 by 16 (left: 8 8 16)
3412 2 * 14 = 28, replace 2 14 by 28 (left: 8 8 28)
3413 2 / 14 = 0.14, replace 2 14 by 0.14 (left: 8 8 0.14)
3414 14 / 2 = 7, replace 2 14 by 7 (left: 8 8 7)
3415 14 - 2 = 12, replace 2 14 by 12 (left: 8 8 12)
3416 2 - 14 = -12, replace 2 14 by -12 (left: 8 8 -12)
3417 Input: 1 (2) 7 (10)
3418 2 + 7 = 9, replace 2 7 by 9 (left: 1 9 10)
3419 2 * 7 = 14, replace 2 7 by 14 (left: 1 14 10)
3420 2 / 7 = 0.29, replace 2 7 by 0.29 (left: 1 0.29 10)
3421 7 / 2 = 3.5, replace 2 7 by 3.5 (left: 1 3.5 10)
3422 7 - 2 = 5, replace 2 7 by 5 (left: 1 5 10)
3423 2 - 7 = -5, replace 2 7 by -5 (left: 1 -5 10)
3424 Input: (7) (8) 9
3425 7 + 8 = 15, replace 7 8 by 15 (left: 15 9)
3426 7 * 8 = 56, replace 7 8 by 56 (left: 56 9)
3427 7 / 8 = 0.88, replace 7 8 by 0.88 (left: 0.88 9)
3428 8 / 7 = 1.14, replace 7 8 by 1.14 (left: 1.14 9)
3429 8 - 7 = 1, replace 7 8 by 1 (left: 1 9)
3430 7 - 8 = -1, replace 7 8 by -1 (left: -1 9)
3431 Input: {input}
3432 '''
3433
3434 propose_prompt_5s = '''Use the two numbers in the bracket and
3435 basic arithmetic operations to propose possible next steps.
3436 Then, remove the selected numbers by the new number. Use the
3437 format '(left: ...)' to present the remaining numbers.
3438 Do not include serial numbers that are not in the example. Do
3439 not include unnecessary information in your output.
3440 Input: (2) 8 8 (14)
3441 2 + 14 = 16, replace 2 14 by 16 (left: 8 8 16)
3442 2 * 14 = 28, replace 2 14 by 28 (left: 8 8 28)
3443 2 / 14 = 0.14, replace 2 14 by 0.14 (left: 8 8 0.14)
3444 14 / 2 = 7, replace 2 14 by 7 (left: 8 8 7)
3445 14 - 2 = 12, replace 2 14 by 12 (left: 8 8 12)
3446 2 - 14 = -12, replace 2 14 by -12 (left: 8 8 -12)
3447 Input: 1 (2) 7 (10)
3448 2 + 7 = 9, replace 2 7 by 9 (left: 1 9 10)
3449 2 * 7 = 14, replace 2 7 by 14 (left: 1 14 10)
3450 2 / 7 = 0.29, replace 2 7 by 0.29 (left: 1 0.29 10)
3451 7 / 2 = 3.5, replace 2 7 by 3.5 (left: 1 3.5 10)
3452 7 - 2 = 5, replace 2 7 by 5 (left: 1 5 10)
3453 2 - 7 = -5, replace 2 7 by -5 (left: 1 -5 10)
3454 Input: (7) (8) 9
3455 7 + 8 = 15, replace 7 8 by 15 (left: 15 9)
3456 7 * 8 = 56, replace 7 8 by 56 (left: 56 9)
3457 7 / 8 = 0.88, replace 7 8 by 0.88 (left: 0.88 9)
3458 8 / 7 = 1.14, replace 7 8 by 1.14 (left: 1.14 9)
3459 8 - 7 = 1, replace 7 8 by 1 (left: 1 9)
3460 7 - 8 = -1, replace 7 8 by -1 (left: -1 9)
3461 Input: (2.33) (6)
3462 2.33 + 6 = 8.33, replace 2.33 6 by 8.33 (left: 8.33)
3463 2.33 * 6 = 14, replace 2.33 6 by 14 (left: 14)
3464 2.33 / 6 = 0.39, replace 2.33 6 by 0.39 (left: 0.39)
3465 6 / 2.33 = 2.57, replace 2.33 6 by 2.57 (left: 2.57)
3466 6 - 2.33 = 3.67, replace 2.33 6 by 3.67 (left: 3.67)
3467 2.33 - 6 = -3.67, replace 2.33 6 by -3.67 (left: -3.67)
3468 Input: 0.66 (8) (9)
3469 8 + 9 = 17, replace 8 9 by 17 (left: 0.66 17)
3470 8 * 9 = 72, replace 8 9 by 72 (left: 0.66 72)
3471 8 / 9 = 0.89, replace 8 9 by 0.89 (left: 0.66 0.89)
3472 9 / 8 = 1.12, replace 8 9 by 1.12 (left: 0.66 1.12)
3473 9 - 8 = 1, replace 8 9 by 1 (left: 0.66 1)
3474 8 - 9 = -1, replace 8 9 by -1 (left: 0.66 -1)
3475 Input: {input}
3476 '''

assembly_prompt_1s = '''Use the previous steps of equations to
form a final equation that obtains 24. Use 'Answer: ' to
present your final answer.
Input: 4 4 6 8
Steps:
4 + 8 = 12 (left: 4 6 12)
6 - 4 = 2 (left: 2 12)
2 * 12 = 24 (left: 24)
Let's do it step by step:
f1 = 4 + 8 = 12. In this step, 4 and 8 are from the input.
f2 = 6 - 4 = 2. In this step, 6 and 4 are from the input.
f3 = 2 * 12 = 24. In this step, 2 is from f2, and 12 is from
f1.
Thus, we replace 2 by f2: f3 = 2 * 12 = f2 * 12 = 24
Thus, we replace 12 by f1: f3 = 2 * 12 = f2 * f1 = 24
Since f1 = 4 + 8, we replace f1 by 4 + 8: f3 = 2 * 12 = f2 *
(4 + 8) = 24
Since f2 = 6 - 4, we replace f2 by 6 - 4: f3 = 2 * 12 = (6 -
4) * (4 + 8) = 24
Answer: (6 - 4) * (4 + 8) = 24
Input: {input}Let's do it step by step:
f1 = '''
assembly_prompt_3s = '''Use the previous steps of equations to
form a final equation that obtains 24. Use 'Answer: ' to
present your final answer.
Input: 4 4 6 8
Steps:
4 + 8 = 12 (left: 4 6 12)
6 - 4 = 2 (left: 2 12)
2 * 12 = 24 (left: 24)
Let's do it step by step:
f1 = 4 + 8 = 12. In this step, 4 and 8 are from the input.
f2 = 6 - 4 = 2. In this step, 6 and 4 are from the input.
f3 = 2 * 12 = 24. In this step, 2 is from f2, and 12 is from
f1.
Thus, we replace 2 by f2: f3 = 2 * 12 = f2 * 12 = 24
Thus, we replace 12 by f1: f3 = 2 * 12 = f2 * f1 = 24
Since f1 = 4 + 8, we replace f1 by 4 + 8: f3 = 2 * 12 = f2 *
(4 + 8) = 24
Since f2 = 6 - 4, we replace f2 by 6 - 4: f3 = 2 * 12 = (6 -
4) * (4 + 8) = 24
Answer: (6 - 4) * (4 + 8) = 24
Input: 2 9 10 12
Steps:
12 * 2 = 24 (left: 9 10 24)
10 - 9 = 1 (left: 1 24)
24 * 1 = 24 (left: 24)
Let's do it step by step:
f1 = 12 * 2 = 24. In this step, 12 and 2 are from the input.
f2 = 10 - 9 = 1. In this step, 10 and 9 are from the input.
f3 = 24 * 1 = 24. In this step, 24 is from f1, and 1 is from
f2.
Thus, we replace 24 by f1: f3 = 24 * 1 = f1 * 1 = 24
Thus, we replace 1 by f2: f3 = 24 * 1 = f1 * f2 = 24
Since f1 = 12 * 2, we replace f1 by 12 * 2: f3 = 24 * 1 = (12
* 2) * f2 = 24
Since f2 = 10 - 9, we replace f2 by 10 - 9: f3 = 24 * 1 = (12
* 2) * (10 - 9) = 24
Answer: (12 * 2) * (10 - 9) = 24
Input: 4 9 10 13
Steps:
13 - 10 = 3 (left: 3 4 9)
9 - 3 = 6 (left: 4 6)
4 * 6 = 24 (left: 24)
Let's do it step by step:
f1 = 13 - 10 = 3. In this step, 13 and 10 are from the input.
f2 = 9 - 3 = 6. In this step, 9 is from the input, and 3 is
from f1.
Thus, we replace 3 by f1: f2 = 9 - 3 = 9 - f1 = 6
f3 = 4 * 6 = 24. In this step, 4 is from the input, and 6 is
from f2.
Thus, we replace 6 by f2: f3 = 4 * 6 = 4 * f2 = 24
Since f2 = 9 - f1, we replace f2 by 9 - f1: f3 = 4 * 6 = 4 *
(9 - f1) = 24
Since f1 = 13 - 10, we replace f1 by 13 - 10: f3 = 4 * 6 = 4 *
(9 - (13 - 10)) = 24
Answer: 4 * (9 - (13 - 10)) = 24
Input: {input}Let's do it step by step:
f1 = '''
assembly_prompt_5s = '''Use the previous steps of equations to
form a final equation that obtains 24. Use 'Answer: ' to
present your final answer.
Input: 4 4 6 8
Steps:

3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563

3564 4 + 8 = 12 (left: 4 6 12)
3565 6 - 4 = 2 (left: 2 12)
3566 2 * 12 = 24 (left: 24)
3567 Let's do it step by step:
3568 f1 = 4 + 8 = 12. In this step, 4 and 8 are from the input.
3569 f2 = 6 - 4 = 2. In this step, 6 and 4 are from the input.
3570 f3 = 2 * 12 = 24. In this step, 2 is from f2, and 12 is from
3571 f1.
3572 Thus, we replace 2 by f2: f3 = 2 * 12 = f2 * 12 = 24
3573 Thus, we replace 12 by f1: f3 = 2 * 12 = f2 * f1 = 24
3574 Since f1 = 4 + 8, we replace f1 by 4 + 8: f3 = 2 * 12 = f2 *
3575 (4 + 8) = 24
3576 Since f2 = 6 - 4, we replace f2 by 6 - 4: f3 = 2 * 12 = (6 -
3577 4) * (4 + 8) = 24
3578 Answer: (6 - 4) * (4 + 8) = 24
3579 Input: 2 9 10 12
3580 Steps:
3581 12 * 2 = 24 (left: 9 10 24)
3582 10 - 9 = 1 (left: 1 24)
3583 24 * 1 = 24 (left: 24)
3584 Let's do it step by step:
3585 f1 = 12 * 2 = 24. In this step, 12 and 2 are from the input.
3586 f2 = 10 - 9 = 1. In this step, 10 and 9 are from the input.
3587 f3 = 24 * 1 = 24. In this step, 24 is from f1, and 1 is from
3588 f2.
3589 Thus, we replace 24 by f1: f3 = 24 * 1 = f1 * 1 = 24
3590 Thus, we replace 1 by f2: f3 = 24 * 1 = f1 * f2 = 24
3591 Since f1 = 12 * 2, we replace f1 by 12 * 2: f3 = 24 * 1 = (12
3592 * 2) * f2 = 24
3593 Since f2 = 10 - 9, we replace f2 by 10 - 9: f3 = 24 * 1 = (12
3594 * 2) * (10 - 9) = 24
3595 Answer: (12 * 2) * (10 - 9) = 24
3596 Input: 4 9 10 13
3597 Steps:
3598 13 - 10 = 3 (left: 3 4 9)
3599 9 - 3 = 6 (left: 4 6)
3600 4 * 6 = 24 (left: 24)
3601 Let's do it step by step:
3602 f1 = 13 - 10 = 3. In this step, 13 and 10 are from the input.
3603 f2 = 9 - 3 = 6. In this step, 9 is from the input, and 3 is
3604 from f1.
3605 Thus, we replace 3 by f1: f2 = 9 - 3 = 9 - f1 = 6
3606 f3 = 4 * 6 = 24. In this step, 4 is from the input, and 6 is
3607 from f2.
3608 Thus, we replace 6 by f2: f3 = 4 * 6 = 4 * f2 = 24
3609 Since f2 = 9 - f1, we replace f2 by 9 - f1: f3 = 4 * 6 = 4 *
3610 (9 - f1) = 24
3611 Since f1 = 13 - 10, we replace f1 by 13 - 10: f3 = 4 * 6 = 4 *
3612 (9 - (13 - 10)) = 24
3613 Answer: 4 * (9 - (13 - 10)) = 24
3614 Input: 1 4 8 8
3615 Steps:
3616 8 / 4 = 2 (left: 1 2 8)
3617 1 + 2 = 3 (left: 3 8)
3618 3 * 8 = 24 (left: 24)
3619 Let's do it step by step:
3620 f1 = 8 / 4 = 2. In this step, 8 and 4 are from the input.
3621 f2 = 1 + 2 = 3. In this step, 2 is from f1, and 1 is from the
3622 input.
3623 Thus, we replace 2 by f1: f2 = 1 + 2 = 1 + f1 = (1 + (8 / 4))
3624 = 3
3625 f3 = 3 * 8 = 24. In this step, 3 is from f2, and 8 is from the
3626 input.
3627 Thus, we replace 3 by f2: f3 = 3 * 8 = f2 * 8 = 24
3628 Since f2 = 1 + f1, we replace f2 by 1 + f1: f3 = 3 * 8 = (1 +
3629 f1) * 8 = 24
3630 Since f1 = 8 / 4, we replace f1 by 8 / 4: (1 + f1) * 8 = (1 +
3631 (8 / 4)) * 8 = 24
3632 Answer: (1 + (8 / 4)) * 8 = 24
3633 Input: 5 5 5 9
3634 Steps:
3635 5 + 5 = 10 (left: 5 9 10)
3636 10 + 5 = 15 (left: 9 15)
3637 15 + 9 = 24 (left: 24)
3638 Let's do it step by step:
3639 f1 = 5 + 5 = 10. In this step, 5 and 5 are from the input.
3640 f2 = 10 + 5 = 15. In this step, 10 is from f1, and 5 is from
3641 the input.
3642 Thus, we replace 10 by f1: f2 = 10 + 5 = f1 + 5 = 15
3643 f3 = 15 + 9 = 24. In this step, 15 is from f2, and 9 is from
3644 the input.
3645 Thus, we replace 15 by f2: f3 = 15 + 9 = f2 + 9 = 24
3646 Since f2 = f1 + 5, we replace f2 by f1 + 5: f3 = 15 + 9 = (f1
3647 + 5) + 9 = 24
3648 Since f1 = 5 + 5, we replace f1 by 5 + 5: f3 = 15 + 9 = ((5 +
3649 5) + 9) = 24
3650 Answer: ((5 + 5) + 9) = 24

Input: {input}Let's do it step by step:
f1 = ''

3651
3652

Listing 17: MusiQue prompts

direct = ""Please answer a question given some paragraphs as
context. Respond only the answer, in the format "The answer is
###."

3653
3654
3655

Below are some examples (contexts are omitted):

3656

Question: What is the extreme low temperature of the city
where WNJN-FM is located?
The answer is -9 °F.

3657
3658
3659
3660

Question: When did muslim armies invade the country Al-Mahabah
is located and the country Kleicha originates?
The answer is in 634.

3661
3662
3663
3664

Question: When did hurricane Sandy his the city where The
Dealer's performer was born?
The answer is October 28, 2012.

3665
3666
3667
3668

Question: What is the enrollment of undergraduates at the
university attended by the entrepreneur owning the gold spike
in the location holding PollyGrind Film Festival?
The answer is 7,200.

3669
3670
3671
3672
3673

Question: When did the nation that seized the country where Al-
Berka is located from the empire that declined following the
Crimean War join the Allies in WW2?
The answer is September 1943.

3674
3675
3676
3677
3678

Question: An institution like a German Fachhochschule is known
by what term in Éric Losfeld's birth country and the country
where painters remained focused on textures and surfaces.
The answer is hogeschool.

3679
3680
3681
3682
3683

Now, answer a question given the following paragraphs as
context:

3684
3685

{ctx}

3686

Question: {question}
The answer is""

3687
3688
3689

cot = ""Please answer a question given some paragraphs as
context. Respond by decomposing the question into subquestions,
and end with the format "The answer is ###."

3690
3691
3692
3693

Below are some examples (contexts are omitted):

3694

Question: What is the extreme low temperature of the city
where WNJN-FM is located?

3695
3696

The question can be decomposed into following subquestions:

3697

What city is WNJN-FM located? Answer: Atlantic City

3700

What is the extreme low temperature of Atlantic City ? Answer:

3701

-9 °F

3702

The answer is -9 °F.

3703
3704

Question: When did muslim armies invade the country Al-Mahabah
is located and the country Kleicha originates?

3705
3706
3707

The question can be decomposed into following subquestions:

3708

Which was the country for Kleicha? Answer: Iraq

3709

Which country is Al-Mahabah in? Answer: Syria

3710

When did muslim armies invade Syria and Iraq? Answer: in 634

3711

The answer is in 634.

3712
3713

Question: When did hurricane Sandy his the city where The
Dealer's performer was born?

3714
3715
3716

The question can be decomposed into following subquestions:

3717

Who is The Dealers' performer? Answer: Mal Waldron

3718

Where is Mal Waldron's place of birth? Answer: New York City

3719

When did hurricane sandy hit New York City? Answer: October 28,

3720

2012

3721

The answer is October 28, 2012.

3722
3723

Question: What is the enrollment of undergraduates at the
university attended by the entrepreneur owning the gold spike
in the location holding PollyGrind Film Festival?

3724
3725

The question can be decomposed into following subquestions:

3726

What is the location of PollyGrind Film Festival? Answer: Las

3727

Vegas

3728

Who owns the gold spike in Las Vegas? Answer: Tony Hsieh

3729

Tony Hsieh is educated at where? Answer: Harvard

3730

What is the enrollment of undergraduates at Harvard ? Answer:

3731

7,200

3732
3733

3734	The answer is 7,200.	(Éric Losfeld, place of birth, Mouscron)	3821
3735		(The Collegian, owned by, Houston Baptist University)	3822
3736	Question: When did the nation that seized the country where Al-	(Houston Baptist University, founded, 1960)	3823
3737	Berka is located from the empire that declined following the	(Hertfordshire, located in, East of England)	3824
3738	Crimean War join the Allies in WW2?	(Jan Šindel, birthplace, Hradec Králové)	3825
3739	The question can be decomposed into following subquestions:	(Arrondissement of Mouscron, country, Belgium)	3826
3740	Which country is Al-Berka in? Answer: Libya	(The focus of paintings on textures and surfaces, located in,	3827
3741	What empire declined after the Crimean War? Answer: the	the Netherlands)	3828
3742	Ottoman Empire	(Belgium and the Netherlands, refer to, hogeschool)	3829
3743	Which european state seized Libya from the Ottoman Empire in	Question: An institution like a German Fachhochschule is known	3830
3744	1911? Answer: Italy	by what term in Éric Losfeld's birth country and the country	3831
3745	When did Italy join the allies in ww2? Answer: September 1943	where painters remained focused on textures and surfaces.	3832
3746	The answer is September 1943.	The answer is hogeschool.	3833
3747			3834
3748	Question: An institution like a German Fachhochschule is known	Now, answer a question given the following relation triplets	3835
3749	by what term in Éric Losfeld's birth country and the country	as context:	3836
3750	where painters remained focused on textures and surfaces.		3837
3751	The question can be decomposed into following subquestions:	{ctx}	3838
3752	Where is Éric Losfeld's place of birth? Answer: Mouscron	Question: {question}	3839
3753	Arrondissement of Mouscron >> country? Answer: Belgium	The answer is ""	3840
3754	Where was the focus of paintings on textures and surfaces?		3841
3755	Answer: the Netherlands	cot_triplet = ""Please answer a question given some relation	3842
3756	What term is used in Belgium and the the Netherlands to refer	triplets as context. Respond by decomposing the question into	3843
3757	to an institution like a German Fachhochschule? Answer:	subquestions, and end with the format "The answer is ###."	3844
3758	hogeschool		3845
3759	The answer is hogeschool.		3846
3760			3847
3761	Now, answer a question given the following paragraphs as	Below are some examples:	3848
3762	context:	Relation treiplets:	3849
3763		(WNJN-FM, located in, Atlantic City)	3850
3764	{ctx}	(The extreme low temperature of Atlantic City, is, -9 °F)	3851
3765		Question: What is the extreme low temperature of the city	3852
3766	Question: {question}	where WNJN-FM is located?	3853
3767	The question can be decomposed into following subquestions: ""	The question can be decomposed into following subquestions:	3854
3768		What city is WNJN-FM located? Answer: Atlantic City	3855
3769		What is the extreme low temperature of Atlantic City ? Answer:	3856
3770	direct_triplet = ""Please answer a question given some	-9 °F	3857
3771	relation triplets as context. Respond only the answer, in the	The answer is -9 °F.	3858
3772	format "The answer is ###."		3859
3773	Below are some examples:	Relation treiplets:	3860
3774		(Kleicha, country, Iraq)	3861
3775	Relation triplets:	(Al-Mahabah, country, Syria)	3862
3776	(WNJN-FM, located in, Atlantic City)	(Muslim armies, invade, Syria and Iraq)	3863
3777	(The extreme low temperature of Atlantic City, is, -9 °F)	(Syria and Iraq, invaded in, 634)	3864
3778	Question: What is the extreme low temperature of the city	Question: When did muslim armies invade the country Al-Mahabah	3865
3779	where WNJN-FM is located?	is located and the country Kleicha originates?	3866
3780	The answer is -9 °F.	The question can be decomposed into following subquestions:	3867
3781		Which was the country for Kleicha? Answer: Iraq	3868
3782	Relation triplets:	Which country is Al-Mahabah in? Answer: Syria	3869
3783	(Kleicha, country, Iraq)	When did muslim armies invade Syria and Iraq? Answer: in 634	3870
3784	(Al-Mahabah, country, Syria)	The answer is in 634.	3871
3785	(Muslim armies, invade, Syria and Iraq)		3872
3786	(Syria and Iraq, invaded in, 634)	Relation treiplets:	3873
3787	Question: When did muslim armies invade the country Al-Mahabah	(The Dealers, performer, Mal Waldron)	3874
3788	is located and the country Kleicha originates?	(Mal Waldron, place of birth, New York City)	3875
3789	The answer is in 634.	(Hurricane Sandy, hit New York City, October 28, 2012)	3876
3790		Question: When did hurricane Sandy his the city where The	3877
3791	Relation triplets:	Dealer's performer was born?	3878
3792	(The Dealers, performer, Mal Waldron)	The question can be decomposed into following subquestions:	3879
3793	(Mal Waldron, place of birth, New York City)	Who is The Dealers' performer? Answer: Mal Waldron	3880
3794	(Hurricane Sandy, hit New York City, October 28, 2012)	Where is Mal Waldron's place of birth? Answer: New York City	3881
3795	Question: When did hurricane Sandy his the city where The	When did hurricane sandy hit New York City? Answer: October 28,	3882
3796	Dealer's performer was born?	2012	3883
3797	The answer is October 28, 2012.	The answer is October 28, 2012.	3884
3798			3885
3799	Relation triplets:		3886
3800	(PollyGrind Film Festival, location, Las Vegas)		3887
3801	(Gold Spike, owned by, Tony Hsieh)	Relation treiplets:	3888
3802	(Tony Hsieh, educated at, Harvard)	(PollyGrind Film Festival, location, Las Vegas)	3889
3803	(Harvard, undergraduate enrollment, 7,200)	(Gold Spike, owned by, Tony Hsieh)	3890
3804	Question: What is the enrollment of undergraduates at the	(Tony Hsieh, educated at, Harvard)	3891
3805	university attended by the entrepreneur owning the gold spike	(Harvard, undergraduate enrollment, 7,200)	3892
3806	in the location holding PollyGrind Film Festival?	Question: What is the enrollment of undergraduates at the	3893
3807	The answer is 7,200.	university attended by the entrepreneur owning the gold spike	3894
3808		in the location holding PollyGrind Film Festival?	3895
3809	Relation triplets:	The question can be decomposed into following subquestions:	3896
3810	(Al-Berka, country, Libya)	What is the location of PollyGrind Film Festival? Answer: Las	3897
3811	(Ottoman Empire, declined after, Crimean War)	Vegas	3898
3812	(The European state, seized Libya from the Ottoman Empire in	Who owns the gold spike in Las Vegas? Answer: Tony Hsieh	3899
3813	1911, Italy)	Tony Hsieh is educated at where? Answer: Harvard	3900
3814	(Italy, join the allies in WW2, September 1943)	What is the enrollment of undergraduates at Harvard ? Answer:	3901
3815	Question: When did the nation that seized the country where Al-	7,200	3902
3816	Berka is located from the empire that declined following the	The answer is 7,200.	3903
3817	Crimean War join the Allies in WW2?		3904
3818	The answer is September 1943.		3905
3819			3906
3820	Relation triplets:	Relation treiplets:	3907
		(Al-Berka, country, Libya)	

3908	(Ottoman Empire, declined after, Crimean War)	The answer to the original question is October 28, 2012.	3995
3909	(The European state, seized Libya from the Ottoman Empire in		3996
3910	1911, Italy)	Question: What is the enrollment of undergraduates at the	3997
3911	(Italy, join the allies in WW2, September 1943)	university attended by the entrepreneur owning the gold spike	3998
3912	Question: When did the nation that seized the country where Al-	in the location holding PollyGrind Film Festival?	3999
3913	Berka is located from the empire that declined following the	The question can be decomposed into following subquestions:	4000
3914	Crimean War join the Allies in WW2?	sub question: What is the location of PollyGrind Film Festival	4001
3915	The question can be decomposed into following subquestions:	?	4002
3916	Which country is Al-Berka in? Answer: Libya	sub answer: Las Vegas	4003
3917	What empire declined after the Crimean War? Answer: the	sub question: Who owns the gold spike in Las Vegas?	4004
3918	Ottoman Empire	sub answer: Tony Hsieh	4005
3919	Which european state seized Libya from the Ottoman Empire in	sub question: Tony Hsieh is educated at where?	4006
3920	1911? Answer: Italy	sub answer: Harvard	4007
3921	When did Italy join the allies in ww2? Answer: September 1943	sub question: What is the enrollment of undergraduates at	4008
3922	The answer is September 1943.	Harvard ?	4009
3923		sub answer: 7,200	4010
3924		The answer to the original question is 7,200.	4011
3925	Relation triplets:		4012
3926	(Éric Losfeld, place of birth, Mouscron)	Question: When did the nation that seized the country where Al-	4013
3927	(The Collegian, owned by, Houston Baptist University)	Berka is located from the empire that declined following the	4014
3928	(Houston Baptist University, founded, 1960)	Crimean War join the Allies in WW2?	4015
3929	(Hertfordshire, located in, East of England)	The question can be decomposed into following subquestions:	4016
3930	(Jan Šindel, birthplace, Hradec Králové)	sub question: Which country is Al-Berka in?	4017
3931	(Arrondissement of Mouscron, country, Belgium)	sub answer: Libya	4018
3932	(The focus of paintings on textures and surfaces, located in,	sub question: What empire declined after the Crimean War?	4019
3933	the Netherlands)	sub answer: the Ottoman Empire	4020
3934	(Belgium and the Netherlands, refer to, hogeschool)	sub question: Which european state seized Libya from the	4021
3935	Question: An institution like a German Fachhochschule is known	Ottoman Empire in 1911?	4022
3936	by what term in Éric Losfeld's birth country and the country	sub answer: Italy	4023
3937	where painters remained focused on textures and surfaces.	sub question: When did Italy join the allies in ww2?	4024
3938	The question can be decomposed into following subquestions:	sub answer: September 1943	4025
3939	Where is Éric Losfeld's place of birth? Answer: Mouscron	The answer to the original question is September 1943.	4026
3940	Arrondissement of Mouscron >> country? Answer: Belgium		4027
3941	Where was the focus of paintings on textures and surfaces?	Question: An institution like a German Fachhochschule is known	4028
3942	Answer: the Netherlands	by what term in Éric Losfeld's birth country and the country	4029
3943	What term is used in Belgium and the the Netherlands to refer	where painters remained focused on textures and surfaces.	4030
3944	to an institution like a German Fachhochschule? Answer:	The question can be decomposed into following subquestions:	4031
3945	hogeschool	sub question: Where is Éric Losfeld's place of birth?	4032
3946	The answer is hogeschool.	sub answer: Mouscron	4033
3947		sub question: Arrondissement of Mouscron >> country?	4034
3948		sub answer: Belgium	4035
3949	Now, answer a question given the following relation triplets	sub question: Where was the focus of paintings on textures and	4036
3950	as context:	surfaces?	4037
3951		sub answer: the Netherlands	4038
3952	{ctx}	sub question: What term is used in Belgium and the the	4039
3953		Netherlands to refer to an institution like a German	4040
3954	Question: {question}	Fachhochschule?	4041
3955	The question can be decomposed into following subquestions:""	sub answer: hogeschool	4042
3956		The answer to the original question is hogeschool.	4043
3957	tot = ""Please answer a question given some paragraphs as		4044
3958	context. Respond by decomposing the question into subquestions,	Now, answer a question given the following paragraphs as	4045
3959	and end with the format "The answer to the original question	context. Remember to decompose the question into subquestions,	4046
3960	is ###." All questions can find their answers in the provided	and it is guaranteed that the answer can be found in the	4047
3961	paragraphs.	provided paragraphs:	4048
3962			4049
3963	Below are some examples to demonstrate the desired answering	{ctx}	4050
3964	format (contexts are omitted for brevity):		4051
3965	Question: What is the extreme low temperature of the city	Now answer the following question. Remember to decompose the	4052
3966	where WNJN-FM is located?	question into subquestions, and it is guaranteed that the	4053
3967	The question can be decomposed into following subquestions:	answer can be found in the provided paragraphs.	4054
3968	sub question: What city is WNJN-FM located?		4055
3969	sub answer: Atlantic City	Question: {question}	4056
3970	sub question: What is the extreme low temperature of Atlantic	The question can be decomposed into following subquestions:""	4057
3971	City ?		4058
3972	sub answer: -9 °F	tot_triplet = ""Please answer a question given some	4059
3973	The answer to the original question is -9 °F.	paragraphs as context. Respond by decomposing the question	4060
3974		into subquestions, and end with the format "The answer to the	4061
3975	Question: When did muslim armies invade the country Al-Mahabab	original question is ###." All questions can find their	4062
3976	is located and the country Kleicha originates?	answers in the provided paragraphs.	4063
3977	The question can be decomposed into following subquestions:		4064
3978	sub question: Which was the country for Kleicha?	Below are some examples to demonstrate the desired answering	4065
3979	sub answer: Iraq	format (contexts are omitted for brevity):	4066
3980	sub question: Which country is Al-Mahabab in?	Question: What is the extreme low temperature of the city	4067
3981	sub answer: Syria	where WNJN-FM is located?	4068
3982	sub question: When did muslim armies invade Syria and Iraq?	The question can be decomposed into following subquestions:	4069
3983	sub answer: in 634	sub question: What city is WNJN-FM located?	4070
3984	The answer to the original question is in 634.	sub answer: Atlantic City	4071
3985		sub question: What is the extreme low temperature of Atlantic	4072
3986	Question: When did hurricane Sandy his the city where The	City ?	4073
3987	Dealer's performer was born?	sub answer: -9 °F	4074
3988	The question can be decomposed into following subquestions:	The answer to the original question is -9 °F.	4075
3989	sub question: Who is The Dealers' performer?		4076
3990	sub answer: Mal Waldron	Question: When did muslim armies invade the country Al-Mahabab	4077
3991	sub question: Where is Mal Waldron's place of birth?	is located and the country Kleicha originates?	4078
3992	sub answer: New York City	The question can be decomposed into following subquestions:	4079
3993	sub question: When did hurricane sandy hit New York City?	sub question: Which was the country for Kleicha?	4080
3994	sub answer: October 28, 2012	sub answer: Iraq	4081

4082 sub question: Which country is Al-Mahabah in?
4083 sub answer: Syria
4084 sub question: When did muslim armies invade Syria and Iraq?
4085 sub answer: in 634
4086 The answer to the original question is in 634.
4087
4088 Question: When did hurricane Sandy his the city where The
4089 Dealer's performer was born?
4090 The question can be decomposed into following subquestions:
4091 sub question: Who is The Dealers' performer?
4092 sub answer: Mal Waldron
4093 sub question: Where is Mal Waldron's place of birth?
4094 sub answer: New York City
4095 sub question: When did hurricane sandy hit New York City?
4096 sub answer: October 28, 2012
4097 The answer to the original question is October 28, 2012.
4098
4099 Question: What is the enrollment of undergraduates at the
4100 university attended by the entrepreneur owning the gold spike
4101 in the location holding PollyGrind Film Festival?
4102 The question can be decomposed into following subquestions:
4103 sub question: What is the location of PollyGrind Film Festival
4104 ?
4105 sub answer: Las Vegas
4106 sub question: Who owns the gold spike in Las Vegas?
4107 sub answer: Tony Hsieh
4108 sub question: Tony Hsieh is educated at where?
4109 sub answer: Harvard
4110 sub question: What is the enrollment of undergraduates at
4111 Harvard ?
4112 sub answer: 7,200
4113 The answer to the original question is 7,200.
4114
4115 Question: When did the nation that seized the country where Al-
4116 Berka is located from the empire that declined following the
4117 Crimean War join the Allies in WW2?
4118 The question can be decomposed into following subquestions:
4119 sub question: Which country is Al-Berka in?
4120 sub answer: Libya
4121 sub question: What empire declined after the Crimean War?
4122 sub answer: the Ottoman Empire
4123 sub question: Which european state seized Libya from the
4124 Ottoman Empire in 1911?
4125 sub answer: Italy
4126 sub question: When did Italy join the allies in ww2?
4127 sub answer: September 1943
4128 The answer to the original question is September 1943.
4129
4130 Question: An institution like a German Fachhochschule is known
4131 by what term in Éric Losfeld's birth country and the country
4132 where painters remained focused on textures and surfaces.
4133 The question can be decomposed into following subquestions:
4134 sub question: Where is Éric Losfeld's place of birth?
4135 sub answer: Mouscron
4136 sub question: Arrondissement of Mouscron >> country?
4137 sub answer: Belgium
4138 sub question: Where was the focus of paintings on textures and
4139 surfaces?
4140 sub answer: the Netherlands
4141 sub question: What term is used in Belgium and the the
4142 Netherlands to refer to an institution like a German
4143 Fachhochschule?
4144 sub answer: hogeschool
4145 The answer to the original question is hogeschool.
4146
4147 Now, answer a question given the following paragraphs as
4148 context. Remember to decompose the question into subquestions,
4149 and it is guaranteed that the answer can be found in the
4150 provided paragraphs:
4151
4152 {triplet_ctx}
4153
4154 Now answer the following question. Remember to decompose the
4155 question into subquestions, and it is guaranteed that the
4156 answer can be found in the provided paragraphs.
4157
4158 Question: {question}
4159 The question can be decomposed into following subquestions:"""
4160
4161 tot_self_eval = ""
4162 The following are candidates to answer a multi-hop question.
4163 Some of them are complete reasoning trajectories while other
4164 may be intermediate. Please sort these candidates based on how
4165 likely they will lead to a correct solution. You should sort
4166 based on the quality, instead of length, i.e. complete
4167 reasoning may or may not be correct. You should return a comma
4168 separated list, and use #1 to indicate the first candidate,

#2 to indicate the second candidate, and so on. Note that the
multi-hop question can be answered by decomposing them into
subquestions and answering them one by one. A good candidate
should be on the right track of decomposing question, and make
no mistakes, and a good final answer should be as concise as
possible.
Below is an example:
Candidate 1:
Question: Where was the film The Beach filmed in the country
where Pao Sarasin was born?
The question can be decomposed into following subquestions:
sub question: Where was Pao Sarasin born?
sub answer: Bangkok, Thailand
sub question: Where was the film The Beach filmed?
sub answer: The film The Beach was filmed on the Thai island
Koh Phi Phi.
The answer to the original question is Thailand.
Candidate 2:
Question: Where was the film The Beach filmed in the country
where Pao Sarasin was born?
The question can be decomposed into following subquestions:
sub question: Where was Pao Sarasin born?
sub answer: Bangkok, Thailand
sub question: Where was the film The Beach filmed?
sub answer: Koh Phi Phi, Thailand
The answer to the original question is Koh Phi Phi, Thailand.
Candidate 3:
Question: Where was the film The Beach filmed in the country
where Pao Sarasin was born?
The question can be decomposed into following subquestions:
sub question: Where was Pao Sarasin born?
sub answer: Bangkok, Thailand
sub question: Where was the film The Beach filmed?
sub answer: The film The Beach was filmed on the Thai island
Koh Phi Phi, in Thailand.
The answer to the original question is The film The Beach was
filmed on the Thai island Koh Phi Phi, in Thailand.
Candidate 4:
Question: Where was the film The Beach filmed in the country
where Pao Sarasin was born?
The question can be decomposed into following subquestions:
sub question: What is the country of birth of Pao Sarasin?
sub answer: Thailand
sub question: Where was the film The Beach filmed?
sub answer: The Thai island Koh Phi Phi
The answer to the original question is The Thai island Koh Phi
Phi.
Candidate 5:
Question: Where was the film The Beach filmed in the country
where Pao Sarasin was born?
The question can be decomposed into following subquestions:
sub question: What is the country of birth of Pao Sarasin?
sub answer: Thailand
sub question: Where was the film The Beach filmed?
sub answer: The film The Beach was filmed on the Thai island
Koh Phi Phi.
The answer to the original question is The film The Beach was
filmed on the Thai island Koh Phi Phi.
Now pick the most likely candidate to answer the original
question.
Answer: #4, #5, #3, #2, #1
Now pick from the following candidates:
{candidates}
Now pick the most likely candidate to answer the original
question.
Answer:
"""

4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4240
4241
4242
4243
4244
4245

Listing 18: Blocksworld prompts

```
{
  "example_pool": [
    {
      "init": "the red block is clear, the orange block
is clear, the hand is empty, the orange block is on top of the
blue block, the red block is on the table and the blue block
is on the table",
```

4246
4247
4248
4249
4250
4251
4252

```

4253     "goal": "the blue block is on top of the orange
4254 block",
4255     "plan": "\nunstack the orange block from on top of
4256 the blue block\nput down the orange block\npick up the blue
4257 block\nstack the blue block on top of the orange block\n[PLAN
4258 END]\n",
4259     "states": [
4260         "the red block is clear, the blue block is
4261 clear, the orange block is clear, the hand is empty, the red
4262 block is on the table, the blue block is on the table and the
4263 orange block is on the table",
4264         "the red block is clear, the blue block is
4265 clear, the hand is empty, the blue block is on top of the
4266 orange block, the red block is on the table and the orange
4267 block is on the table"
4268     ]
4269 },
4270 {
4271     "init": "the blue block is clear, the orange block
4272 is clear, the hand is empty, the red block is on top of the
4273 yellow block, the orange block is on top of the red block, the
4274 blue block is on the table and the yellow block is on the
4275 table",
4276     "goal": "the blue block is on top of the yellow
4277 block and the orange block is on top of the blue block",
4278     "plan": "\nunstack the orange block from on top of
4279 the red block\nput down the orange block\nunstack the red
4280 block from on top of the yellow block\nput down the red block\n
4281 npick up the blue block\nstack the blue block on top of the
4282 yellow block\npick up the orange block\nstack the orange block
4283 on top of the blue block\n[PLAN END]\n",
4284     "states": [
4285         "the red block is clear, the blue block is
4286 clear, the orange block is clear, the hand is empty, the red
4287 block is on top of the yellow block, the blue block is on the
4288 table, the orange block is on the table and the yellow block
4289 is on the table",
4290         "the red block is clear, the blue block is
4291 clear, the orange block is clear, the yellow block is clear,
4292 the hand is empty, the red block is on the table, the blue
4293 block is on the table, the orange block is on the table and
4294 the yellow block is on the table",
4295         "the red block is clear, the blue block is
4296 clear, the orange block is clear, the hand is empty, the blue
4297 block is on top of the yellow block, the red block is on the
4298 table, the orange block is on the table and the yellow block
4299 is on the table",
4300         "the red block is clear, the orange block is
4301 clear, the hand is empty, the blue block is on top of the
4302 yellow block, the orange block is on top of the blue block,
4303 the red block is on the table and the yellow block is on the
4304 table"
4305     ]
4306 },
4307 {
4308     "init": "the red block is clear, the yellow block
4309 is clear, the hand is empty, the red block is on top of the
4310 blue block, the blue block is on top of the orange block, the
4311 orange block is on the table and the yellow block is on the
4312 table",
4313     "goal": "the blue block is on top of the orange
4314 block and the yellow block is on top of the red block",
4315     "plan": "\npick up the yellow block\nstack the
4316 yellow block on top of the red block\n[PLAN END]\n",
4317     "states": [
4318         "the yellow block is clear, the hand is empty,
4319 the red block is on top of the blue block, the blue block is
4320 on top of the orange block, the yellow block is on top of the
4321 red block and the orange block is on the table"
4322     ]
4323 },
4324 {
4325     "init": "the blue block is clear, the yellow block
4326 is clear, the hand is empty, the red block is on top of the
4327 orange block, the blue block is on top of the red block, the
4328 orange block is on the table and the yellow block is on the
4329 table",
4330     "goal": "the blue block is on top of the red block
4331 and the yellow block is on top of the blue block",
4332     "plan": "\npick up the yellow block\nstack the
4333 yellow block on top of the blue block\n[PLAN END]\n",
4334     "states": [
4335         "the yellow block is clear, the hand is empty,
4336 the red block is on top of the orange block, the blue block
4337 is on top of the red block, the yellow block is on top of the
4338 blue block and the orange block is on the table"
4339     ]

```

```

    },
    {
        "init": "the blue block is clear, the orange block
4340 is clear, the hand is empty, the blue block is on top of the
4341 yellow block, the yellow block is on top of the red block, the
4342 red block is on the table and the orange block is on the
4343 table",
4344     "goal": "the blue block is on top of the red block
4345 and the orange block is on top of the yellow block",
4346     "plan": "\nunstack the blue block from on top of
4347 the yellow block\nstack the blue block on top of the orange
4348 block\nunstack the yellow block from on top of the red block\n
4349 put down the yellow block\nunstack the blue block from on top
4350 of the orange block\nstack the blue block on top of the red
4351 block\npick up the orange block\nunstack the orange block on top
4352 of the yellow block\n[PLAN END]\n",
4353     "states": [
4354         "the blue block is clear, the yellow block is
4355 clear, the hand is empty, the blue block is on top of the
4356 orange block, the yellow block is on top of the red block, the
4357 red block is on the table and the orange block is on the
4358 table",
4359         "the red block is clear, the blue block is
4360 clear, the yellow block is clear, the hand is empty, the blue
4361 block is on top of the orange block, the red block is on the
4362 table, the orange block is on the table and the yellow block
4363 is on the table",
4364         "the blue block is clear, the orange block is
4365 clear, the yellow block is clear, the hand is empty, the blue
4366 block is on top of the red block, the red block is on the
4367 table, the orange block is on the table and the yellow block
4368 is on the table",
4369         "the blue block is clear, the orange block is
4370 clear, the hand is empty, the blue block is on top of the red
4371 block, the orange block is on top of the yellow block, the red
4372 block is on the table and the yellow block is on the table"
4373     ]
4374 },
4375 {
4376     "init": "the blue block is clear, the orange block
4377 is clear, the hand is empty, the blue block is on top of the
4378 red block, the red block is on the table and the orange block
4379 is on the table",
4380     "goal": "the red block is on top of the orange
4381 block and the orange block is on top of the blue block",
4382     "plan": "\nunstack the blue block from on top of
4383 the red block\nput down the blue block\npick up the orange
4384 block\nstack the orange block on top of the blue block\npick
4385 up the red block\nstack the red block on top of the orange
4386 block\n[PLAN END]\n",
4387     "states": [
4388         "the red block is clear, the blue block is
4389 clear, the orange block is clear, the hand is empty, the red
4390 block is on the table, the blue block is on the table and the
4391 orange block is on the table",
4392         "the red block is clear, the orange block is
4393 clear, the hand is empty, the orange block is on top of the
4394 blue block, the red block is on the table and the blue block
4395 is on the table",
4396         "the red block is clear, the hand is empty,
4397 the red block is on top of the orange block, the orange block
4398 is on top of the blue block and the blue block is on the table"
4399     ]
4400 },
4401 {
4402     "init": "the red block is clear, the yellow block
4403 is clear, the hand is empty, the red block is on top of the
4404 orange block, the orange block is on top of the blue block,
4405 the blue block is on the table and the yellow block is on the
4406 table",
4407     "goal": "the red block is on top of the yellow
4408 block, the blue block is on top of the orange block and the
4409 yellow block is on top of the blue block",
4410     "plan": "\nunstack the red block from on top of
4411 the orange block\nput down the red block\nunstack the orange
4412 block from on top of the blue block\nput down the orange block\n
4413 pick up the blue block\nstack the blue block on top of the
4414 orange block\npick up the yellow block\nstack the yellow block
4415 on top of the blue block\npick up the red block\nstack the
4416 red block on top of the yellow block\n[PLAN END]\n",
4417     "states": [
4418         "the red block is clear, the orange block is
4419 clear, the yellow block is clear, the hand is empty, the
4420 orange block is on top of the blue block, the red block is on
4421 the table, the blue block is on the table and the yellow block
4422 is on the table",

```

```

4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4420
4421
4422
4423
4424
4425
4426

```

```

4427         "the red block is clear, the blue block is
4428 clear, the orange block is clear, the yellow block is clear,
4429 the hand is empty, the red block is on the table, the blue
4430 block is on the table, the orange block is on the table and
4431 the yellow block is on the table",
4432         "the red block is clear, the blue block is
4433 clear, the yellow block is clear, the hand is empty, the blue
4434 block is on top of the orange block, the red block is on the
4435 table, the orange block is on the table and the yellow block
4436 is on the table",
4437         "the red block is clear, the yellow block is
4438 clear, the hand is empty, the blue block is on top of the
4439 orange block, the yellow block is on top of the blue block,
4440 the red block is on the table and the orange block is on the
4441 table",
4442         "the red block is clear, the hand is empty,
4443 the red block is on top of the yellow block, the blue block is
4444 on top of the orange block, the yellow block is on top of the
4445 blue block and the orange block is on the table"
4446     ]
4447 },
4448 {
4449     "init": "the red block is clear, the blue block is
4450 clear, the hand is empty, the red block is on top of the
4451 orange block, the blue block is on the table and the orange
4452 block is on the table",
4453     "goal": "the red block is on top of the blue block
4454 and the blue block is on top of the orange block",
4455     "plan": "\nunstack the red block from on top of
4456 the orange block\nput down the red block\npick up the blue
4457 block\nstack the blue block on top of the orange block\npick
4458 up the red block\nstack the red block on top of the blue block
4459 \n[PLAN END]\n",
4460     "states": [
4461         "the red block is clear, the blue block is
4462 clear, the orange block is clear, the hand is empty, the red
4463 block is on the table, the blue block is on the table and the
4464 orange block is on the table",
4465         "the red block is clear, the blue block is
4466 clear, the hand is empty, the blue block is on top of the
4467 orange block, the red block is on the table and the orange
4468 block is on the table",
4469         "the red block is clear, the hand is empty,
4470 the red block is on top of the blue block, the blue block is
4471 on top of the orange block and the orange block is on the
4472 table"
4473     ]
4474 },
4475 {
4476     "init": "the blue block is clear, the yellow block
4477 is clear, the hand is empty, the blue block is on top of the
4478 orange block, the yellow block is on top of the red block, the
4479 red block is on the table and the orange block is on the
4480 table",
4481     "goal": "the blue block is on top of the red block
4482 and the yellow block is on top of the orange block",
4483     "plan": "\nunstack the blue block from on top of
4484 the orange block\nput down the blue block\nunstack the yellow
4485 block from on top of the red block\nstack the yellow block on
4486 top of the orange block\npick up the blue block\nstack the
4487 blue block on top of the red block\n[PLAN END]\n",
4488     "states": [
4489         "the blue block is clear, the orange block is
4490 clear, the yellow block is clear, the hand is empty, the
4491 yellow block is on top of the red block, the red block is on
4492 the table, the blue block is on the table and the orange block
4493 is on the table",
4494         "the red block is clear, the blue block is
4495 clear, the yellow block is clear, the hand is empty, the
4496 yellow block is on top of the orange block, the red block is
4497 on the table, the blue block is on the table and the orange
4498 block is on the table",
4499         "the blue block is clear, the yellow block is
4500 clear, the hand is empty, the blue block is on top of the red
4501 block, the yellow block is on top of the orange block, the red
4502 block is on the table and the orange block is on the table"
4503     ]
4504 },
4505 {
4506     "init": "the red block is clear, the orange block
4507 is clear, the white block is clear, the hand is empty, the
4508 blue block is on top of the yellow block, the white block is
4509 on top of the blue block, the red block is on the table, the
4510 orange block is on the table and the yellow block is on the
4511 table",
4512     "goal": "the blue block is on top of the yellow
4513 block, the orange block is on top of the white block and the

```

```

white block is on top of the red block",
    "plan": "\nunstack the white block from on top of
the blue block\nstack the white block on top of the red block\n
pick up the orange block\nstack the orange block on top of
the white block\n[PLAN END]\n",
    "states": [
        "the blue block is clear, the orange block is
clear, the white block is clear, the hand is empty, the blue
block is on top of the yellow block, the white block is on top
of the red block, the red block is on the table, the orange
block is on the table and the yellow block is on the table",
        "the blue block is clear, the orange block is
clear, the hand is empty, the blue block is on top of the
yellow block, the orange block is on top of the white block,
the white block is on top of the red block, the red block is
on the table and the yellow block is on the table"
    ]
},
    "intro": "I am playing with a set of blocks where I need
to arrange the blocks into stacks. Here are the actions I can
do\n\nPick up a block\nUnstack a block from on top of another
block\nPut down a block\nStack a block on top of another block
\n\nI have the following restrictions on my actions:\nI can
only pick up or unstack one block at a time.\nI can only pick
up or unstack a block if my hand is empty.\nI can only pick up
a block if the block is on the table and the block is clear.
A block is clear if the block has no other blocks on top of it
and if the block is not picked up.\nI can only unstack a
block from on top of another block if the block I am
unstacking was really on top of the other block.\nI can only
unstack a block from on top of another block if the block I am
unstacking is clear.\nOnce I pick up or unstack a block, I am
holding the block.\nI can only put down a block that I am
holding.\nI can only stack a block on top of another block if
I am holding the block being stacked.\nI can only stack a
block on top of another block if the block onto which I am
stacking the block is clear.\nOnce I put down or stack a block,
my hand becomes empty.\n\n",
    "world_update_pickup": "I am playing with a set of blocks
where I need to arrange the blocks into stacks. Here are the
actions I can do \n\nPick up a block \nUnstack a block from on
top of another block \nPut down a block \nStack a block on
top of another block \n\nI have the following restrictions on
my actions:\nI can only pick up or unstack one block at a time.
\nI can only pick up or unstack a block if my hand is empty.
\nI can only pick up a block if the block is on the table and
the block is clear. A block is clear if the block has no other
blocks on top of it and if the block is not picked up. \nI
can only unstack a block from on top of another block if the
block I am unstacking was really on top of the other block. \n
I can only unstack a block from on top of another block if
the block I am unstacking is clear. Once I pick up or unstack
a block, I am holding the block. \nI can only put down a block
that I am holding. \nI can only stack a block on top of
another block if I am holding the block being stacked. \nI can
only stack a block on top of another block if the block onto
which I am stacking the block is clear. Once I put down or
stack a block, my hand becomes empty.\n\nAfter being given an
initial state and an action, give the new state after
performing the action.\n\n[SCENARIO 1]\n\n[STATE 0] I have that,
the white block is clear, the cyan block is clear, the brown
block is clear, the hand is empty, the white block is on top
of the purple block, the purple block is on the table, the
cyan block is on the table and the brown block is on the table
.\n\n[ACTION] Pick up the brown block.\n\n[CHANGE] The hand was
empty and is now holding the brown block, the brown block was
on the table and is now in the hand, and the brown block is no
longer clear.\n\n[STATE 1] I have that, the white block is
clear, the cyan block is clear, the brown block is in the hand,
the hand is holding the brown block, the white block is on
top of the purple block, the purple block is on the table and
the cyan block is on the table.\n\n[SCENARIO 2]\n\n[STATE 0] I
have that, the purple block is clear, the cyan block is clear,
the white block is clear, the hand is empty, the white block
is on top of the brown block, the purple block is on the table,
the cyan block is on the table and the brown block is on the
table.\n\n[ACTION] Pick up the cyan block.\n\n[CHANGE] The hand
was empty and is now holding the cyan block, the cyan block
was on the table and is now in the hand, and the cyan block is
no longer clear.\n\n[STATE 1] I have that, the cyan block is in
the hand, the white block is clear, the purple block is clear,
the hand is holding the cyan block, the white block is on top
of the brown block, the purple block is on the table and the
brown block is on the table.\n\n[SCENARIO 3]\n\n[STATE 0] I have
that, \n\n[ACTION] \n\n[CHANGE]",
    "world_update_unstack": "I am playing with a set of blocks

```

```

4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600

```

4601 where I need to arrange the blocks into stacks. Here are the
4602 actions I can do \n\nPick up a block \nUnstack a block from on
4603 top of another block \nPut down a block \nStack a block on
4604 top of another block \n\nI have the following restrictions on
4605 my actions:\nI can only pick up or unstack one block at a time.
4606 \nI can only pick up or unstack a block if my hand is empty.
4607 \nI can only pick up a block if the block is on the table and
4608 the block is clear. A block is clear if the block has no other
4609 blocks on top of it and if the block is not picked up. \nI
4610 can only unstack a block from on top of another block if the
4611 block I am unstacking was really on top of the other block. \n
4612 \nI can only unstack a block from on top of another block if
4613 the block I am unstacking is clear. Once I pick up or unstack
4614 a block, I am holding the block. \nI can only put down a block
4615 that I am holding. \nI can only stack a block on top of
4616 another block if I am holding the block being stacked. \nI can
4617 only stack a block on top of another block if the block onto
4618 which I am stacking the block is clear. Once I put down or
4619 stack a block, my hand becomes empty.\n\nAfter being given an
4620 initial state and an action, give the new state after
4621 performing the action.\n\n[SCENARIO 1]\n[STATE 0] I have that,
4622 the white block is clear, the cyan block is clear, the brown
4623 block is clear, the hand is empty, the white block is on top
4624 of the purple block, the purple block is on the table, the
4625 cyan block is on the table and the brown block is on the table
4626 .\n\n[ACTION] Unstack the white block from on top of the purple
4627 block.\n\n[CHANGE] The hand was empty and is now holding the
4628 white block, the white block was on top of the purple block
4629 and is now in the hand, the white block is no longer clear,
4630 and the purple block is now clear.\n\n[STATE 1] I have that, the
4631 purple block is clear, the cyan block is clear, the brown
4632 block is clear, the hand is holding the white block, the white
4633 block is in the hand, the purple block is on the table, the
4634 cyan block is on the table and the brown block is on the table
4635 .\n\n[SCENARIO 2]\n[STATE 0] I have that, the purple block is
4636 clear, the cyan block is clear, the white block is clear, the
4637 hand is empty, the cyan block is on top of the brown block,
4638 the purple block is on the table, the white block is on the
4639 table and the brown block is on the table.\n\n[ACTION] Unstack
4640 the cyan block from on top of the brown block.\n\n[CHANGE] The
4641 hand was empty and is now holding the cyan block, the cyan
4642 block was on top of the brown block and is now in the hand,
4643 the cyan block is no longer clear, and the brown block is now
4644 clear.\n\n[STATE 1] I have that, the purple block is clear, the
4645 brown block is clear, the cyan block is in the hand, the white
4646 block is clear, the hand is holding the cyan block, the
4647 purple block is on the table, the white block is on the table
4648 and the brown block is on the table.\n\n[SCENARIO 3]\n[STATE
4649 0] I have that, {} \n\n[ACTION] {} \n\n[CHANGE]”,
4650 "world_update_putdown": "I am playing with a set of blocks
4651 where I need to arrange the blocks into stacks. Here are the
4652 actions I can do \n\nPick up a block \nUnstack a block from on
4653 top of another block \nPut down a block \nStack a block on
4654 top of another block \n\nI have the following restrictions on
4655 my actions:\nI can only pick up or unstack one block at a time.
4656 \nI can only pick up or unstack a block if my hand is empty.
4657 \nI can only pick up a block if the block is on the table and
4658 the block is clear. A block is clear if the block has no other
4659 blocks on top of it and if the block is not picked up. \nI
4660 can only unstack a block from on top of another block if the
4661 block I am unstacking was really on top of the other block. \n
4662 \nI can only unstack a block from on top of another block if
4663 the block I am unstacking is clear. Once I pick up or unstack
4664 a block, I am holding the block. \nI can only put down a block
4665 that I am holding. \nI can only stack a block on top of
4666 another block if I am holding the block being stacked. \nI can
4667 only stack a block on top of another block if the block onto
4668 which I am stacking the block is clear. Once I put down or
4669 stack a block, my hand becomes empty.\n\nAfter being given an
4670 initial state and an action, give the new state after
4671 performing the action.\n\n[SCENARIO 1]\n[STATE 0] I have that,
4672 the white block is clear, the purple block is clear, the cyan
4673 block is in the hand, the brown block is clear, the hand is
4674 holding the cyan block, the white block is on the table, the
4675 purple block is on the table, and the brown block is on the
4676 table.\n\n[ACTION] Put down the cyan block.\n\n[CHANGE] The hand
4677 was holding the cyan block and is now empty, the cyan block
4678 was in the hand and is now on the table, and the cyan block is
4679 now clear.\n\n[STATE 1] I have that, the cyan block is clear,
4680 the purple block is clear, the white block is clear, the brown
4681 block is clear, the hand is empty, the white block is on the
4682 table, the purple block is on the table, the cyan block is on
4683 the table, and the brown block is on the table.\n\n[SCENARIO
4684 2]\n[STATE 0] I have that, the purple block is clear, the
4685 black block is in the hand, the white block is clear, the hand
4686 is holding the black block, the white block is on top of the
4687 brown block, the purple block is on the table, and the brown

4688 block is on the table.\n\n[ACTION] Put down the black block.\n\n[
4689 CHANGE] The hand was holding the black block and is now empty,
4690 the black block was in the hand and is now on the table, and
4691 the black block is now clear.\n\n[STATE 1] I have that, the
4692 black block is clear, the purple block is clear, the white
4693 block is clear, the hand is empty, the white block is on top
4694 of the brown block, the purple block is on the table, the
4695 brown block is on the table, and the black block is on the
4696 table.\n\n[SCENARIO 3]\n[STATE 0] I have that, {} \n\n[ACTION]
4697 {} \n\n[CHANGE]”,
4698 "world_update_stack": "I am playing with a set of blocks
4699 where I need to arrange the blocks into stacks. Here are the
4700 actions I can do \n\nPick up a block \nUnstack a block from on
4701 top of another block \nPut down a block \nStack a block on
4702 top of another block \n\nI have the following restrictions on
4703 my actions:\nI can only pick up or unstack one block at a time.
4704 \nI can only pick up or unstack a block if my hand is empty.
4705 \nI can only pick up a block if the block is on the table and
4706 the block is clear. A block is clear if the block has no other
4707 blocks on top of it and if the block is not picked up. \nI
4708 can only unstack a block from on top of another block if the
4709 block I am unstacking was really on top of the other block. \n
4710 \nI can only unstack a block from on top of another block if
4711 the block I am unstacking is clear. Once I pick up or unstack
4712 a block, I am holding the block. \nI can only put down a block
4713 that I am holding. \nI can only stack a block on top of
4714 another block if I am holding the block being stacked. \nI can
4715 only stack a block on top of another block if the block onto
4716 which I am stacking the block is clear. Once I put down or
4717 stack a block, my hand becomes empty.\n\nAfter being given an
4718 initial state and an action, give the new state after
4719 performing the action.\n\n[SCENARIO 1]\n[STATE 0] I have that,
4720 the white block is clear, the purple block is clear, the cyan
4721 block is in the hand, the brown block is clear, the hand is
4722 holding the cyan block, the white block is on the table, the
4723 purple block is on the table, and the brown block is on the
4724 table.\n\n[ACTION] Stack the cyan block on top of the brown
4725 block.\n\n[CHANGE] The hand was holding the cyan block and is
4726 now empty, the cyan block was in the hand and is now on top of
4727 the brown block, the brown block is no longer clear, and the
4728 cyan block is now clear.\n\n[STATE 1] I have that, the cyan
4729 block is clear, the purple block is clear, the white block is
4730 clear, the hand is empty, the cyan block is on top of the
4731 brown block, the brown block is on the table, the purple block
4732 is on the table, and the white block is on the table.\n\n[
4733 SCENARIO 2]\n[STATE 0] I have that, the purple block is clear,
4734 the black block is in the hand, the white block is clear, the
4735 hand is holding the black block, the white block is on top of
4736 the brown block, the purple block is on the table, and the
4737 brown block is on the table.\n\n[ACTION] Stack the black block
4738 on top of the purple block.\n\n[CHANGE] The hand was holding the
4739 black block and is now empty, the black block was in the hand
4740 and is now on top of the purple block, the purple block is no
4741 longer clear, and the black block is now clear.\n\n[STATE 1] I
4742 have that, the black block is clear, the white block is clear,
4743 the hand is empty, the black block is on top of the purple
4744 block, the white block is on top of the brown block, the brown
4745 block is on the table, and the purple block is on the table.\n
4746 \n\n[SCENARIO 3]\n[STATE 0] I have that, {} \n\n[ACTION] {} \n\n[
4747 CHANGE]”,
4748 "self-eval": "I am playing with a set of blocks where I
4749 need to arrange the blocks into stacks. Here are the actions I
4750 can do\n\nPick up a block\nUnstack a block from on top of
4751 another block\nPut down a block\nStack a block on top of
4752 another block\n\nI have the following restrictions on my
4753 actions:\nI can only pick up or unstack one block at a time.\n
4754 \nI can only pick up or unstack a block if my hand is empty.\n
4755 \nI can only pick up a block if the block is on the table and the
4756 block is clear. A block is clear if the block has no other
4757 blocks on top of it and if the block is not picked up.\n
4758 \nI can only unstack a block from on top of another block if the block
4759 I am unstacking was really on top of the other block.\n
4760 \nI can only unstack a block from on top of another block if the block
4761 I am unstacking is clear.\n
4762 \nOnce I pick up or unstack a block,
4763 I am holding the block.\n
4764 \nI can only put down a block that I
4765 am holding.\n
4766 \nI can only stack a block on top of another block
4767 if I am holding the block being stacked.\n
4768 \nI can only stack a
4769 block on top of another block if the block onto which I am
4770 stacking the block is clear.\n
4771 \nOnce I put down or stack a block,
4772 my hand becomes empty.\n
4773 \nPlease evaluate whether the given
4774 action is a good one under certain conditions.\n\n[STATEMENT]\n
4775 \nAs initial conditions I have that, the red block is clear,
4776 the yellow block is clear, the hand is empty, the red block is
4777 on top of the blue block, the yellow block is on top of the
4778 orange block, the blue block is on the table and the orange
4779 block is on the table.\n
4780 \nMy goal is to have that the orange
4781 block is on top of the red block.\n\n[ACTION]\nUnstack the red

4775 block from on top of the blue block\n[EVALUATION]\nbad\n\n[
4776 STATEMENT]\nAs initial conditions I have that, the orange
4777 block is in the hand, the yellow block is clear, the hand is
4778 holding the orange block, the blue block is on top of the red
4779 block, the yellow block is on top of the blue block, and the
4780 red block is on the table.\nMy goal is to have have that the
4781 yellow block is on top of the orange block.\n[ACTION]\nput
4782 down the orange block\n[EVALUATION]\ngood\n\n[STATEMENT]\nAs
4783 initial conditions I have that, the orange block is clear, the
4784 yellow block is clear, the hand is empty, the blue block is
4785 on top of the red block, the orange block is on top of the
4786 blue block, the red block is on the table and the yellow block
4787 is on the table.\nMy goal is to have that the blue block is
4788 on top of the red block and the yellow block is on top of the
4789 orange block.\n[ACTION]\npick up the yellow block\n[EVALUATION
4790]\ngood\n\n[STATEMENT]\nAs initial conditions I have that, the
4791 orange block is clear, the yellow block is clear, the hand is
4792 empty, the blue block is on top of the red block, the orange
4793 block is on top of the blue block, the red block is on the
4794 table and the yellow block is on the table.\nMy goal is to
4795 have that the blue block is on top of the red block and the
4796 yellow block is on top of the orange block.\n[ACTION]\npick up
4797 the yellow block\n[EVALUATION]\ngood\n\n[STATEMENT]\nAs
4798 initial conditions I have that, the blue block is clear, the
4799 orange block is in the hand, the red block is clear, the hand
4800 is holding the orange block, the red block is on top of the
4801 yellow block, the blue block is on the table, and the yellow
4802 block is on the table.\nMy goal is to have have that the red
4803 block is on top of the yellow block and the orange block is on
4804 top of the blue block.\n[ACTION]\nstack the orange block on
4805 top of the red block\n[EVALUATION]\nbad\n\n[STATEMENT]\nAs
4806 initial conditions I have that, <init_state>\nMy goal is to <
4807 goals>\n[ACTION]\n<action>\n[EVALUATION]\n",
4808 "action_proposals": "I am playing with a set of blocks
4809 where I need to arrange the blocks into stacks. Here are the
4810 actions I can do \n\nPick up a block \nUnstack a block from on
4811 top of another block \nPut down a block \nStack a block on
4812 top of another block \n\nI have the following restrictions on
4813 my actions:\nI can only pick up or unstack one block at a time.
4814 \nI can only pick up or unstack a block if my hand is empty.
4815 \nI can only pick up a block if the block is on the table and
4816 the block is clear. A block is clear if the block has no other
4817 blocks on top of it and if the block is not picked up. \nI
4818 can only unstack a block from on top of another block if the
4819 block I am unstacking was really on top of the other block. \n
4820 \nI can only unstack a block from on top of another block if
4821 the block I am unstacking is clear. Once I pick up or unstack
4822 a block, I am holding the block. \nI can only put down a block
4823 that I am holding. \nI can only stack a block on top of
4824 another block if I am holding the block being stacked. \nI can
4825 only stack a block on top of another block if the block onto
4826 which I am stacking the block is clear. Once I put down or
4827 stack a block, my hand becomes empty.\n\nAfter being given an
4828 initial state, propose all possible actions that is valid in
4829 the given state.\n\n[SCENARIO 1]\n\n[STATE] I have that, the
4830 white block is clear, the purple block is clear, the cyan
4831 block is in the hand, the brown block is clear, the hand is
4832 holding the cyan block, the white block is on the table, the
4833 purple block is on the table, and the brown block is on the
4834 table.\n[ACTION] Stack the cyan block on top of the white
4835 block. Stack the cyan block on top of the purple block. Stack
4836 the cyan block on top of the brown block. Put down the cyan
4837 block\n\n[SCENARIO 2]\n\n[STATE] I have that, the orange block
4838 is clear, the yellow block is clear, the hand is empty, the
4839 blue block is on top of the red block, the orange block is on
4840 top of the blue block, the red block is on the table and the
4841 yellow block is on the table.\n[ACTION] Unstack the orange
4842 block from on top of the blue block. Pick up the yellow block.
4843 \n\n[SCENARIO 3]\n\n[STATE] I have that, {} \n[ACTION]",
4844 "action_proposals_decomp": "I am playing with a set of
4845 blocks where I need to arrange the blocks into stacks. Here
4846 are the actions I can do \n\nPick up a block \nUnstack a block
4847 from on top of another block \nPut down a block \nStack a
4848 block on top of another block \n\nI have the following
4849 restrictions on my actions:\nI can only pick up or unstack one
4850 block at a time. \nI can only pick up or unstack a block if
4851 my hand is empty. \nI can only pick up a block if the block is
4852 on the table and the block is clear. A block is clear if the
4853 block has no other blocks on top of it and if the block is not
4854 picked up. \nI can only unstack a block from on top of
4855 another block if the block I am unstacking was really on top
4856 of the other block. \nI can only unstack a block from on top
4857 of another block if the block I am unstacking is clear. Once I
4858 pick up or unstack a block, I am holding the block. \nI can
4859 only put down a block that I am holding. \nI can only stack a
4860 block on top of another block if I am holding the block being
4861 stacked. \nI can only stack a block on top of another block if

the block onto which I am stacking the block is clear. Once I
put down or stack a block, my hand becomes empty.\n\nAfter
being given an initial state, proposing all possible actions
that is valid.\n\n[SCENARIO 1]\n\n[STATE] I have that, the white
block is clear, the purple block is clear, the cyan block is
in the hand, the brown block is clear, the hand is holding the
cyan block, the white block is on the table, the purple block
is on the table, and the brown block is on the table.\n[
REASON] Since cyan block is in the hand, I can only stack the
cyan block or put down the cyan block. Since white block is
clear, we can stack cyan block on top of the white block.
Since the purple block is clear, we can stack the cyan block
on top of the purple block. Since the brown block is clear, we
can put the cyan block on top of the brown block. \n[ACTION]
Stack the cyan block on top of the white block. Stack the cyan
block on top of the purple block. Stack the cyan block on top
of the brown block. Put down the cyan block\n\n[SCENARIO 2]\n\n
[STATE] I have that, the orange block is clear, the yellow
block is clear, the hand is empty, the blue block is on top of
the red block, the orange block is on top of the blue block,
the red block is on the table and the yellow block is on the
table.\n[REASON] Since the hand is empty, I can only unstack a
block or pick up a block. Since only the orange block and
yellow block is clear, I can only pick up or unstack the
yellow block and orange block. Since the yellow block is on
the table, I can pick up the yellow block. Since the orange
block is on top of the blue block, I can unstack the orange
block from on top of the blue block. \n[ACTION] Unstack the
orange block from on top of the blue block. Pick up the yellow
block. \n\n[SCENARIO 3]\n\n[STATE] I have that, {} \n[REASON]"
}

4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892