A Comparative Study on Reasoning Patterns of OpenAI's o1 Model

Anonymous ACL submission

Abstract

Enabling Large Language Models (LLMs) to handle a wider range of complex tasks (e.g., coding, math) has drawn great attention from many researchers. As LLMs continue to evolve, increasing the number of model parameters yields diminishing performance improvements and heavy computational costs. Recently, OpenAI's o1 model has shown that inference strategies (i.e., Test-time Compute methods) can also significantly enhance the reasoning capabilities of LLMs. However, the mechanisms behind these methods are still unexplored. In our work, to investigate the reasoning patterns of o1, we compare o1 with existing Test-time Compute methods (BoN, Step-wise BoN, Agent Workflow, and Self-Refine) by using OpenAI's GPT-4o as a backbone on general reasoning benchmarks in three domains (i.e., math, code and commonsense reasoning). Specifically, first, our experiments show that the o1 model has achieved the best performance on most datasets. Second, as for the methods of searching diverse responses (e.g., BoN), we find the reward models' capability and the search space both limit the upper boundary of these methods. Third, as for the methods that break the problem into many sub-problems, the Agent Workflow has achieved better performance than Step-wise BoN due to the domain-specific system prompt for planning better reasoning processes. Fourth, we summarize six reasoning patterns of o1, and provide a detailed analysis across different reasoning benchmarks.

1 Introduction

Large Language Models (LLMs) have achieved great success in various tasks (e.g., Commonsense Reasoning (Yang et al., 2018), Coding (Jain et al., 2024; Chai et al., 2024a), Math (Satpute et al., 2024; Chai et al., 2024b), and Dialogue (Young and Shishido, 2023)). To further improve their performance, researchers have continuously increased the number of model parameters and expanded the training data. However, this method of scaling up the model parameters is reaching a bottleneck, and the efficiency of performance improvement is becoming progressively limited.

Recently, Test-time Compute methods, such as Best-of-N (BoN) and Self-Refine (Madaan et al., 2024), have been proposed to enhance model performance during the inference phase and have shown to be more efficient than simply increasing model parameters. However, there is a lack of research comparing the effectiveness of different Test-time Compute methods across various tasks, which would provide valuable guidance for researchers developing new models. Besides, understanding the inference mechanism of the o1 model is very important to help researchers enhance the capabilities of LLMs.

To address the aforementioned issues, we compare OpenAI's o1 model with various Test-time Compute methods, using GPT-4o as the backbone. According to the OpenAI o1 report [1], the model demonstrates exceptional improvements in areas such as mathematics and coding. Therefore, we select four benchmarks—HotpotQA (Yang et al., 2018), Collie (Yao et al., 2023), USACO (Shi et al., 2024), and AIME [2]—to encompass three key reasoning domains. For certain benchmarks (i.e., HotpotQA and Collie) that are not challenging for current LLMs, we follow the LIME (Zhu et al., 2024) and implement a voting method using four selected models (i.e., Qwen (Bai et al., 2023; Yang et al., 2024), Yi (AI et al., 2024), Llama3 (Dubey et al., 2024), and Claude [3]) to filter out samples that cannot be correctly answered by more than two of the LLMs. Then we select four Test-time Compute methods (including Best-of-N (BoN), Step-wise

[1] https://openai.com/index/introducing-openai-o1-preview/

[2] https://huggingface.co/datasets/AI-MO/aimo-validation-aime

[3] https://claude.ai/

BoN, Agent Workflow, and Self-Refine) as baselines which use the GPT-4o as the backbone. As for BoN and Step-wise BoN, we use GPT-4o as the reward model to select the most suitable responses for a given sample. We directly use the code from the GitHub of the Self-Refine (Madaan et al., 2024). As for the Agent Workflow, we utilize the state-of-the-art agent framework (Zhou et al., 2024) on the HotpotQA and Collie, and we use the GPTs [4] for USACO and AIME.

We have conducted comprehensive experiments on our filtered benchmarks, and we have the following insightful findings:

- The OpenAI's o1 model achieves the best results across almost all benchmarks and demonstrates significant improvements in coding and math tasks using the CoT-based approach.

- The domain-specific system prompt is crucial for Step-wise methods. Specifically, the Agent Workflow method greatly enhances the model's performance and it is relatively close to the o1's performance, while the impact of Step-wise BoN on the model's capabilities is mainly evident in the HotpotQA task. Besides, we assume that the Agent Workflow with a series of domain-specific system prompts can not only reduce unnecessary reasoning steps but also carefully align with the reasoning problems.

- We summarize 6 types of o1 reasoning patterns (i.e., **Systematic Analysis (SA)**, **Method Reuse (MR)**, **Divide and Conquer (DC)**, **Self-Refinement (SR)**, **Context Identification (CI)**, and **Emphasizing Constraints (EC)**) across four benchmarks, and we observe that the most commonly used reasoning patterns in o1 are DC and SR, which might be the key to o1's success. Moreover, the reasoning patterns vary across different tasks. Specifically, for commonsense reasoning tasks, o1 tends to use CI and EC. In contrast, in math and coding tasks, o1 mainly relies on MR and DC.

- We also explore the number of reasoning tokens of o1 across different tasks, and observe that the number of reasoning tokens varies a lot across different tasks.

[4] https://openai.com/index/introducing-gpts/

2 Related Work

2.1 Large Language Models

With the emergence of Transformers (Vaswani, 2017) and the scaling laws (Henighan et al., 2020), the researchers try to scale up the parameters of the generative language model. As a result, OpenAI's GPT series models (Radford, 2018; Radford et al., 2019; Brown, 2020; Achiam et al., 2023) have achieved remarkable success in the NLP field. Inspired by the scaling law, the rapid development of open-source models has also been achieved through scaling up the size of parameters and collecting huge data for pre-training, such as Qwen (Bai et al., 2023; Yang et al., 2024), Yi (AI et al., 2024), Llama (Touvron et al., 2023; Dubey et al., 2024), and Deepseek (Bi et al., 2024). Apart from these, current researchers are meeting the demands of training LLMs by collecting higher-quality instruction data and pre-training data. Moreover, improving the quality of the collected data has also gained significant attention in developing LLMs. However, the approach of enhancing model performance by increasing model parameters and collecting more data is facing a bottleneck (Snell et al., 2024).

2.2 Test Time Compute Methods

Snell et al. (2024) propose that scaling LLMs Test-time Compute optimally can be more effective than scaling model parameters. Besides, there are some methods designed for adapting Test-time Compute to LLMs' reasoning. OpenAI's o1 model [5] is designed to spend more time reasoning before they respond for the sake of obtaining better performance. Wang et al. (2023) conduct a hierarchical hypothesis search to enable inductive reasoning capabilities. Besides, A number of related works have been proposed to augment LLMs with tools with Test-time Compute, which can greatly improve their performance on downstream tasks (Gao et al., 2023; Qin et al., 2023; Qu et al., 2024). Moreover, several works have been proposed to learn thought tokens in an unsupervised manner (Goyal et al., 2023; Zelikman et al., 2024), which enable models to more effectively utilize the Test-time compute with sampling longer sequences. In this work, we explore the performance of OpenAI's o1 model on several common NLP reasoning tasks and investigate the reasoning patterns when compared to some classical Test-time Compute methods.

[5] https://openai.com/o1/

3 Experimental Setup

In order to comprehensively evaluate the capability of OpenAI's o1 model, we select and filter 4 benchmarks covering 3 domains (i.e. Commonsense Reasoning, Math, and Code). Then we provide the results of o1, GPT-4o, and some traditional Test-time Compute methods.

3.1 Benchmarks

Commonsense Reasoning. We select **HotpotQA** (Yang et al., 2018) and **Collie** (Yao et al., 2023) to evaluate the commonsense reasoning ability of LLMs. The HotpotQA mainly focuses on commonsense reasoning, which requires LLMs to use multiple supporting documents to answer. Collie needs LLMs to generate text allowing the specification of rich, compositional constraints with diverse generation levels. Due to the excellent open-ended response generation capabilities of GPT-4o and o1, these models demonstrate relatively strong performance on certain benchmarks, particularly in commonsense reasoning. According to LIME (Zhu et al., 2024), we design a **data filtering** module to show the performance differences among different models. This module involves using four different LLMs (i.e., Llama3-72B (Dubey et al., 2024), Qwen-72B (Bai et al., 2023), Claude to answer each sample in those benchmarks and subsequently filtering out samples that more than two models can answer correctly.

Code. We are using the bronze level of the **USACO** (Shi et al., 2024) competition to test the coding skills of LLMs. The USACO focuses on algorithmic and problem-solving skills. We employ LLMs like Llama3-72B, Qwen-72B, and Claude to solve these problems, selecting only those that prove challenging across multiple models to ensure a rigorous assessment of their coding abilities.

Math. We directly use the AIME [6] benchmark to evaluate the model's math ability, which contains 90 problems from AIME 22, AIME 23, and AIME 24, and have been extracted directly from the AOPS wiki page.

3.2 Baseline methods

We select two powerful closed-source LLMs for evaluation.

[6] https://huggingface.co/datasets/AI-MO/aimo-validation-aime

o1 model. It is designed to spend more time reasoning before they respond, which can reason through complex tasks and solve harder problems than previous models in science, coding, and math.

GPT-4o. It is a multimodal model that integrates text, vision, and audio processing capabilities into a single and unified neural network.

As for Test-time Compute methods, we select four methods based on GPT-4o.

Best-of-N (BoN). It makes LLMs generate multiple N outputs for a given input, and the most suitable response is selected as the output.

Step-wise BoN. It enables LLMs to analyze a problem and break it down into several sub-problems. For each step, the model generates N responses based on the previous sub-problems and answers, and then we use a reward model to select the best response. This process continues iteratively until the final answer to the original problem is obtained.

Self-Refine. It improves initial outputs from LLMs through iterative feedback and refinement (Madaan et al., 2024).

Agent Workflow. LLM agents break down complex tasks into smaller sub-tasks, plan their execution through a structured workflow, and utilize various tools to achieve their goals. For the commonsense reasoning datasets, we leverage the existing state-of-the-art agent framework (Zhou et al., 2023, 2024) for evaluation. For the code and math datasets, we select the top-picked agents from GPTs [7], specifically *code copilot* and *math solver*, respectively.

3.3 Metrics

As for HotpotQA and AIME, we design a rule to determine whether the model-generated response contains the correct answer and use the accuracy of the model's responses as the final score. Regarding Collie, we directly determine whether the model-generated response is correct. As for coding tasks (i.e., USACO), we manually run the LLMs-generated code on the test examples, and regard the code passing the test cases as right.

Setting	Baselines	N	Overall	Commonsense Reasoning HotpotQA	Commonsense Reasoning Collie	Code USACO	Math AIME
Direct	o1-preview	-	34.32	14.59	34.07	**44.60**	44.00
	o1-mini	-	**35.77**	15.32	**53.53**	12.23	**62.00**
	GPT4o	-	18.44	13.14	43.36	5.04	12.22
Test-Time	BoN	4	17.65	13.50	39.82	5.04	12.22
	BoN	8	19.04	**16.42**	38.50	7.91	13.33
	Step-wise BoN	1	6.09	13.50	5.31	0.00	5.56
	Step-wise BoN	4	9.79	15.69	19.55	0.00	7.78
	Self-Refine	3	5.62	13.25	0.00	0.00	9.23
	Agent Workflow	-	24.70	14.96	46.07	22.22	15.56

Table 1: The results of OpenAI's o1 model, GPT4o, and some Test-time Compute methods on our selected four benchmarks (i.e., HotpotQA, Collie, USACO, AIME). The '-' in the table represents that the method does not search the multiple responses for generation. **Direct** refers to having the LLMs generate a response directly from the input text, while **Test-Time** refers to using the Test-time Compute method based on GPT-4o.

4 Results

4.1 Overall Analysis

We conduct various experiments to evaluate the performance of o1 and the Test-time Compute methods. As shown in Table 1, the OpenAI's o1 model achieves the best performance on most benchmarks compared to previous Test-time Compute methods and GPT-4o, particularly in Math and Code tasks. Among those benchmarks, o1's improvement in mathematical and coding tasks is particularly notable compared to other methods, which shows that this thinking-before-reasoning approach is more suitable for complex multi-step reasoning in mathematical and coding tasks. Specifically, the o1-mini surpasses the o1-preview on some tasks, it shows that the reasoning process of o1 does not always lead to better improvements.

The performance improvement from Self-Refine is not significant. On most tasks, Self-Refine shows only a slight improvement compared to GPT-4, and its performance even declines on Collie. For this phenomenon, we assume that LLMs may generate responses that slightly deviate from the required format during the refinement iterations of Self-Refine.

BoN achieves relatively good results on HotpotQA. It demonstrates the necessity of searching for more possible responses during the inference stage by scaling time. However, the performance of BoN on Collie has declined compared to the original GPT-4o. Besides, when N increases, there is a slight degradation in performance. We believe this is due to Collie's strict format requirements, which limit the effectiveness of diverse outputs from LLMs.

The Step-wise BoN is limited by the complex tasks. As for Step-wise BoN, it achieves an excellent result on HotpotQA, which does not have a restriction on output text. However, its performance drops significantly on other complex benchmarks that make Step-wise BoN generate numerous intermediate steps and cannot follow the original question.

Agent Workflow achieves a significant improvement in performance on all benchmarks. The Agent Workflow uses a similar idea to the step-wise BoN that breaks down complex tasks into smaller subtasks, but it designs a series of domain-specific system prompts, which reduces unnecessary long-context reasoning processes. However, there is still a gap between the Agent Workflow and the o1 model, which may be because Agent Workflow explores a less diverse space of responses.

4.2 Analysis of the reasoning pattern of o1

As shown in Table 1, although o1 is generally much better than other models, some Test-time Compute methods can still achieve relatively close results to o1 in certain specific tasks. To this end, we analyze the reasoning patterns of o1 across various tasks and summarize the reasoning patterns across different benchmarks as follows:

- **Systematic Analysis (SA).** Starting from the overall structure of the problem, o1 first an-

[7] https://openai.com/index/introducing-gpts/

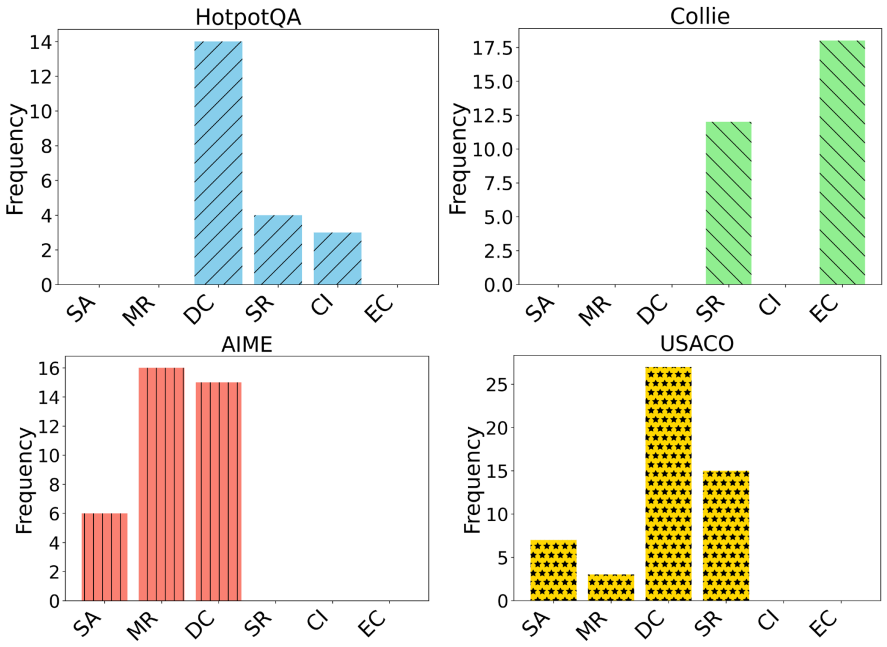

Figure 1: The statistics of different reasoning patterns on different benchmarks.

alyzes the inputs and outputs, as well as the constraints, and then decides on the choice of algorithm and the use of data structures.

- **Method Reuse (MR).** For some problems that can be transformed into classic problems (such as the shortest path or knapsack problem), o1 can quickly reuse existing methods to solve them.

- **Divide and Conquer (DC).** It breaks down a complex problem into subproblems and constructs the overall solution by solving the subproblems.

- **Self-Refinement (SR).** o1 assesses its reasoning process during inference to determine if there are any issues and correct any errors.

- **Context Identification (CI).** For some datasets requiring additional information input (e.g., HotpotQA), o1 first summarizes different aspects of the context related to the query, and then gives the response for the corresponding query.

- **Emphasizing Constraints (EC).** For some datasets with constraints on the generated text (e.g., Collie), o1 usually emphasizes the corresponding constraints during the reasoning process.

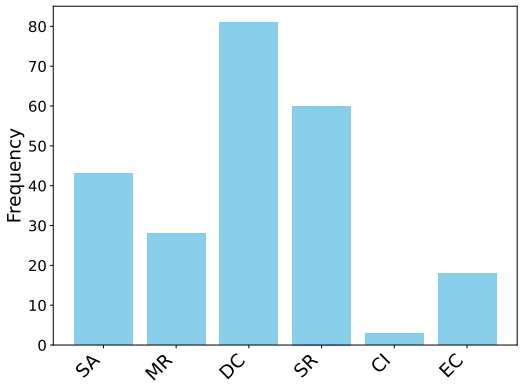

Figure 2: The statistics of reasoning patterns.

We randomly selected 20 to 30 samples of each benchmark to count the number of different reasoning patterns. As shown in Fig. 2, the performance of o1 is primarily influenced by three reasoning patterns: DC, SR, and SA. Among these, SA and DC appear most frequently, suggesting that the combination of SR and DC plays a crucial role in enhancing the performance of o1.

We also show the statistics of the reasoning patterns on different benchmarks in Fig. 1, where different tasks require different reasoning patterns. Specifically, in commonsense reasoning tasks, o1 tends to use task-specific global analysis methods (such as CI and EC) and DC. In math and coding tasks, o1 mainly relies on DC and MR. For both Collie and AIME, o1 follows a relatively shorter

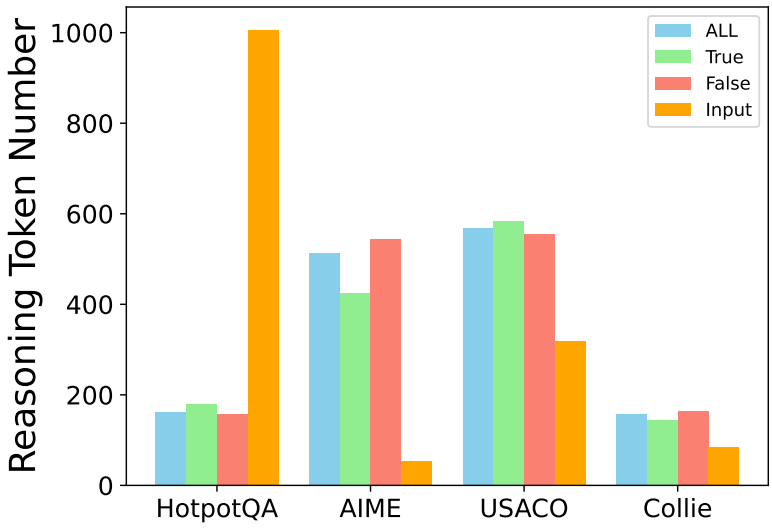

Figure 3: The statistics of the number of o1's reasoning tokens on different tasks. 'ALL' represents the average length of reasoning tokens for all samples, while 'True' and 'False' show the averages for correctly and incorrectly answered samples, respectively. 'Input' refers to the average length of the input prompt.

reasoning process, which we find is also linked to its reasoning patterns. Specifically, o1 often employs the MR approach, where it directly applies well-known classic solutions to solve mathematical problems without the need for multi-step reasoning. In the case of Collie, o1 tends to use the EC reasoning pattern. This allows the model to place greater emphasis on Collie's output format requirements, preventing the generation of an excessively long reasoning process that would result in outputs not meeting the format requirement.

4.3 Long Context Inference Limits Step-Wise BoN

Apart from generating multiple responses in breadth, the Step-wise strategy is also important for scaling inference time. Specifically, the Step-wise methods often produce many intermediate steps, and excessively long context information can prevent the model from following the original input text to generate the correct response. As shown in Table 2, we provide the average number of tokens in the intermediate steps of Step-wise BoN inference across different tasks. The average number of reasoning tokens in almost all tasks exceeded 200, which also confirms that Step-wise BoN requires the model to have strong long-context following capabilities. The Step-wise BoN performs relatively worse on tasks like Collie and AIME, where the output text format and reasoning process are highly complex (for instance, Step-wise BoN achieves less than 12% accuracy on Collie, and its performance

Commonsense Reasoning		**Coding**	**Math**
HotpotQA	Collie	USACO	AIME
273.59	450.31	439.90	262.51

Table 2: The average reasoning token length of Step-wise BoN ($N = 4$).

on AIME is only half that of other methods). However, for tasks (e.g., HotpotQA) that do not require stringent output formatting or intricate reasoning, both BoN and Step-wise BoN significantly enhance the model's results (when $N = 4$, Step-wise BoN outperforms GPT-4o by 2.55% and BoN surpasses GPT-4o by 0.36% on HotpotQA).

4.4 The Number of Reasoning Tokens Across Different Tasks for o1

To investigate whether the number of reasoning tokens is related to o1's ability, we developed a rule to extract o1's reasoning tokens and computed their count across different tasks. Additionally, we calculated the average number of tokens for both correct and incorrect samples. Furthermore, to explore the relation between input prompt length and reasoning tokens length, we also calculate the average input length. As shown in Fig. 3, we observe that the number of reasoning tokens for correct and incorrect samples is similar for the same task, and there is no clear correlation between the input prompt length and the length of the reasoning tokens. Instead, there is a significant difference in reason-

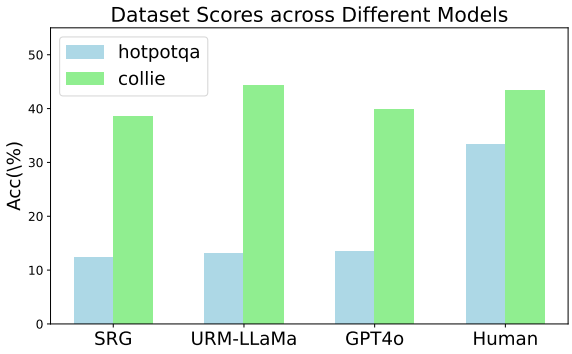

Figure 4: The results of BoN(GPT-4o) using different reward models under $N = 4$ setting. The SRG represents the Skywork-Reward-Gemma-2-27B, the URM-LLaMa refers to the URM-LLaMa-3.1-8B.

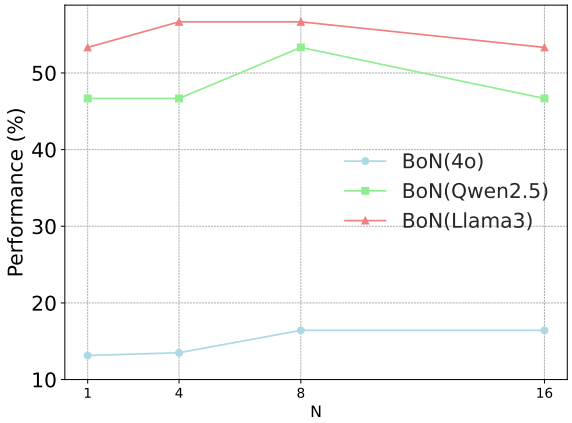

Figure 5: The results of BoN under different search spaces (i.e. the N ranging from 1 to 16) on HotpotQA.

ing tokens across different tasks. Specifically, for commonsense reasoning tasks (i.e., HotpotQA and Collie), the o1's reasoning token length is relatively short. However, for more difficult tasks like Code (i.e., USACO) and Math (i.e., AIME), the model often requires a longer reasoning process to obtain the correct answer.

4.5 The Reward Model Limits Abilities of Searching Methods

As for the BoN series methods, they need to use a reward model to choose the most suitable responses among all the generated responses. Especially for the Step-wise method, an error in any intermediate step can lead to error accumulation, which significantly affects the final output of the model. Therefore, for BoN, we conduct experiments by using different reward models (e.g. Skywork-Reward-Gemma-2-27B (Liu and Zeng, 2024) and URM-LLaMa-3.1-8B (Lou et al., 2024)) from the Leaderboard of RewardBench (Lambert et al., 2024). We

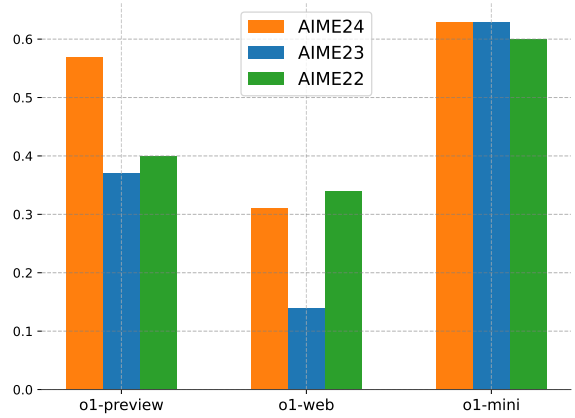

Figure 6: The results of o1 model on AIME24, AIME23, and AIME22.

also use the GPT-4o as the reward model to choose the most suitable response. Moreover, to demonstrate the reward models' limitation on the searching ability of LLMs, we also use the Human as the reward model to judge the most suitable generated response of BoN. As shown in Fig. 4, we provide the results of BoN (GPT-4o) using different reward models. As for HotpotQA, those methods have relatively worse performance (i.e., their accuracies are under 15%), while the human-based reward model could help to improve the LLMs' ACC to 33%. As for the Collie, the performances of using other reward models are close to human results, which show the reward models' ability to determine the upper boundary of those methods. Besides, although the BoN could have comparable results on HotpotQA compared with o1, it can be significantly improved by using a more powerful reward model, demonstrating that the reward model is crucial for the search methods.

4.6 The Search Space also Determines the Upper Boundary of LLMs

Apart from the Agent Workflow, BoN also performs relatively well across various datasets, but its performance is limited by N. To fully explore the upper bound of BoN's capabilities, we increased the value of N in HotpotQA. For the sake of comprehensively evaluating the BoN's capability based on different LLMs with different capability levels, we also evaluate Qwen2.5-72B and Llama3-70B in Fig. 5. Specifically, as shown in Fig. 5, we compare the results of BoN using different backbone models under different search spaces (i.e., $N = 1, 4, 8, 16$). With the N increasing, the performance of BoN tends to stabilize. Notably, both Qwen2.5 and

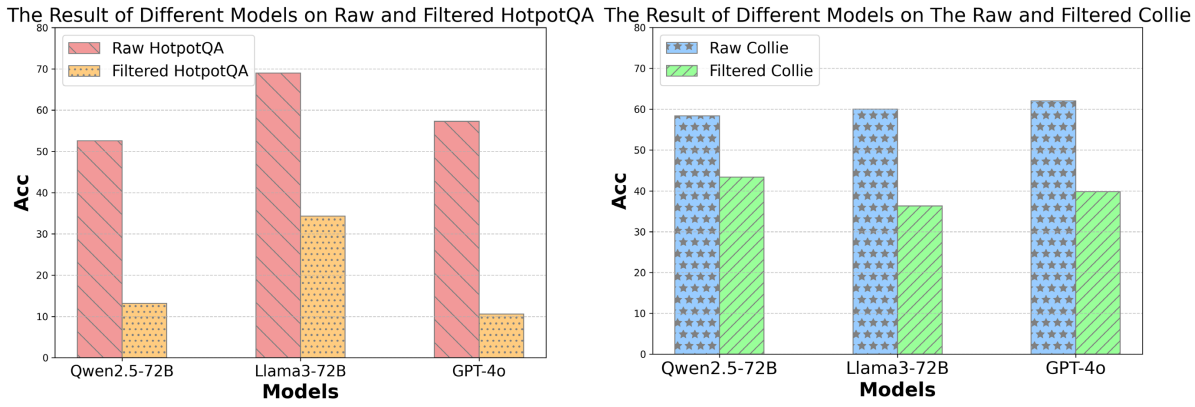

Figure 7: The results of the LLMs on the raw bench and the filtered bench. On the left subfigure, we present LLMs' capabilities on the raw and filtered HotpotQA, and on the right subfigure, we provide the corresponding results on Collie.

Commonsense Reasoning		**Coding**	**Math**
HotpotQA	Collie	USACO	AIME
274	226	139	90

Table 3: The statistics of filtered benchmarks.

Llama3 achieve excellent performance on the HotpotQA dataset. However, when BoN uses these three models as backbone models, performance does not improve consistently with increasing N. When $N > 8$, the performance of the models either stabilizes or declines. We suppose the reason is that the performance of the search methods is jointly related to the reward model and searching space.

4.7 Analysis on Data Filter

The current benchmarks contain many simple samples that cannot distinguish the performance difference across different LLMs, and we filter the samples in HotpotQA and Collie. To demonstrate the impact of our data filter module, we compare the performance of LLMs on benchmarks before and after applying the data filter, where Table 3 presents the statistics about our selected benchmarks after data filtering. As shown in Fig. 7, after data filtering, the scores of different LLMs are relatively lower and show greater distinction. Notably, on HotpotQA, the differences between Qwen2.5 and GPT-4o become evident on our filtered benchmark, which demonstrates the effect of our data filter strategy.

4.8 The Math Ability of OpenAI's o1

To comprehensively evaluate the o1 models' ability in math, we evaluate the o1-preview, o1-web, and o1-mini on the AIME22, AIME23, and AIME24. It is worth mentioning that the o1-mini demonstrates the best performance (around 60%) across these three datasets. However, the performance of the o1-preview fluctuates significantly across different datasets. For example, for o1-preview, the best performance on AIME24 is 57%, while results on the other two datasets are both around 40%.

5 Conclusion

In this work, we explore the capabilities of OpenAI's o1 model in tasks involving mathematics, coding, and commonsense reasoning, and compare o1 with previous Test-time Compute methods (i.e., BoN, Step-wise BoN, Self-Refine, and Agent Workflow), where the findings are as follows. First, we find that o1 achieves better results than other Test-time Compute methods across most tasks. Second, the Agent Workflow method shows significant improvements in all tasks, but the BoN, Step-wise BoN, and Self-Refine methods yield limited improvements due to their reliance on the model's long-context instruction-following ability and the performance of the reward model. Third, we also summarize six reasoning patterns (i.e., SA, MR, DC, SR, CI, EC) of o1, and demonstrate that SR and DC are the main reasoning patterns of o1. We also found that on specific tasks, o1 achieves a higher accuracy by using a certain reasoning pattern, which indicates that these reasoning patterns are crucial for improving reasoning performance. Finally, we hope our study on the reasoning patterns of o1 can guide developers and researchers in understanding the principle of o1 and facilitate the growth of foundation models.

Limitations

In this work, we have extensively investigated the gap between various test-time methods and o1. However, we did not specifically design a reward model to evaluate different paths of reasoning within this chain-of-thought framework, which greatly limited the performance of our baseline in the experiments. Furthermore, although we found that current LLMs struggle to effectively handle longer chains of reasoning, we did not explore whether context compression methods could alleviate this issue.

Ethics Statement

The dataset used in our research is constructed using publicly available data sources, ensuring that there are no privacy concerns or violations. We do not collect any personally identifiable information, and all data used in our research is obtained following legal and ethical standards.

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*.

01. AI, :, Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Tao Yu, Wen Xie, Wenhao Huang, Xiaohui Hu, Xiaoyi Ren, Xinyao Niu, Pengcheng Nie, Yuchi Xu, Yudong Liu, Yue Wang, Yuxuan Cai, Zhenyu Gu, Zhiyuan Liu, and Zonghong Dai. 2024. Yi: Open foundation models by 01.ai. *Preprint*, arXiv:2403.04652.

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

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Linzheng Chai, Shukai Liu, Jian Yang, Yuwei Yin, Ke Jin, Jiaheng Liu, Tao Sun, Ge Zhang, Changyu Ren, Hongcheng Guo, et al. 2024a. Mceval: Massively multilingual code evaluation. *arXiv preprint arXiv:2406.07436*.

Linzheng Chai, Jian Yang, Tao Sun, Hongcheng Guo, Jiaheng Liu, Bing Wang, Xiannian Liang, Jiaqi Bai, Tongliang Li, Qiyao Peng, et al. 2024b. xcot: Cross-lingual instruction tuning for cross-lingual chain-of-thought reasoning. *arXiv preprint arXiv:2401.07037*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2023. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. 2020. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. 2024. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*.

Chris Yuhao Liu and Liang Zeng. 2024. Skywork reward model series. `https://huggingface.co/Skywork`.

Xingzhou Lou, Dong Yan, Wei Shen, Yuzi Yan, Jian Xie, and Junge Zhang. 2024. Uncertainty-aware reward model: Teaching reward models to know what is unknown. *Preprint*, arXiv:2410.00847.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2024. Tool learning with large language models: A survey. *arXiv preprint arXiv:2405.17935*.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Ankit Satpute, Noah Gießing, André Greiner-Petter, Moritz Schubotz, Olaf Teschke, Akiko Aizawa, and Bela Gipp. 2024. Can llms master math? investigating large language models on math stack exchange. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2316–2320.

Quan Shi, Michael Tang, Karthik Narasimhan, and Shunyu Yao. 2024. Can language models solve olympiad programming? *arXiv preprint arXiv:2404.10952*.

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

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

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, and Noah D Goodman. 2023. Hypothesis search: Inductive reasoning with language models. *arXiv preprint arXiv:2309.05660*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Shunyu Yao, Howard Chen, Austin W Hanjie, Runzhe Yang, and Karthik Narasimhan. 2023. Collie: Systematic construction of constrained text generation tasks. *arXiv preprint arXiv:2307.08689*.

Julio Christian Young and Makoto Shishido. 2023. Investigating openai's chatgpt potentials in generating chatbot's dialogue for english as a foreign language learning. *International journal of advanced computer science and applications*, 14(6).

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. 2024. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, et al. 2023. Agents: An open-source framework for autonomous language agents. *arXiv preprint arXiv:2309.07870*.

Wangchunshu Zhou, Yixin Ou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, et al. 2024. Symbolic learning enables self-evolving agents. *arXiv preprint arXiv:2406.18532*.

Kang Zhu, Qianbo Zang, Shian Jia, Siwei Wu, Feiteng Fang, Yizhi Li, Shuyue Guo, Tianyu Zheng, Bo Li, Haoning Wu, et al. 2024. Lime-m: Less is more for evaluation of mllms. *arXiv preprint arXiv:2409.06851*.

A Case Study

The HotpotQA needs LLMs to process multiple documents to obtain the results of the reasoning question, which needs multi-hop reasoning. The o1 generally summarizes the content of the documents from different perspectives to arrive at a solution. As shown in Fig. 8, firstly, the o1 model proposes the "main idea" (i.e., analyzing the context related to the question). Then it obtains the content in the three dimensions (i.e., "Mapping out attractions", "Mapping out attractions", and "Navigating the evolution"). It is worth mentioning that "Navigating the evolution" contains the reasoning chain of the multi-hot reasoning question.

A.1 AIME

Unlike simpler methods that tackle problems through straightforward subtasks, AIME demands a deep integration of diverse mathematical principles to derive accurate solutions. As illustrated in Fig. 9, o1 uses a CoT method following a structured approach: "Identify Key Concepts", "Analyze Constraints", "Apply Mathematical Formulas", and "Construct Logical Reasoning", where the "Logical Reasoning" involves a systematic and detailed process for solving multi-step problems. This reasoning pattern emphasizes the importance of integrating insights from various mathematical concepts to enhance the model's ability to handle complex problems effectively. By following this framework, LLMs can navigate the intricate nature

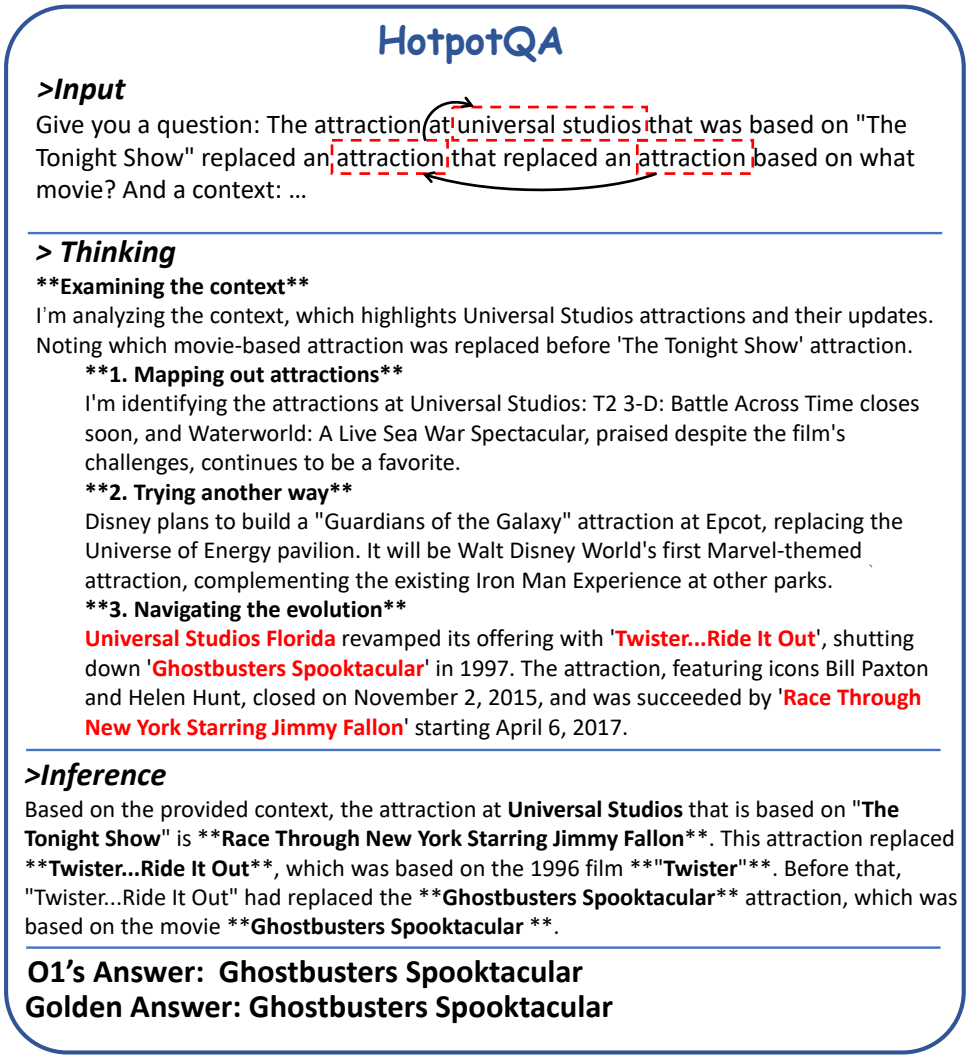

Figure 8: The o1's case of HotpotQA.

of AIME problems, ensuring that all relevant mathematical concepts and logical steps are thoroughly considered.

A.2 Collie

The task of the Collie dataset requires generating a paragraph based on the constraint. Specifically, as illustrated in Fig. 10, at each step of the reasoning process of the o1 model, the form of the constraint is emphasized to strengthen for better generation. Therefore, for tasks requiring strictly controlled generation, the model needs to emphasize the instructions multiple times to guide the content generation.

A.3 USACO

In the USACO competition, o1 demonstrates its robust problem-solving capabilities. As illustrated in Fig. 11, o1 starts by establishing a foundational framework by defining key variables and data structures, and then applies algorithmic logic for state transitions, which will produce the optimal solution gradually. Besides, o1 not only considers all possible paths and scenarios but also uses loops, recursion, and other methods to verify each step rigorously, which helps o1 comprehensively cover multiple aspects of the problems and generate correct solutions efficiently.

AIME

> Input

Let $x_1 \leq x_2 \leq \cdots \leq x_{100}$ be real numbers such that $|x_1| + |x_2| + \cdots + |x_{100}| = 1$ and $x_1 + x_2 + \cdots + x_{100} = 0$. Among all such 100-tuples of numbers, the greatest value that $x_{76} - x_{16}$ can achieve is $\frac{m}{n}$, where m and n are relatively prime positive integers. Find $m + n$.

> Thinking

1.Formulating constraints

I'm mapping out the constraints for $x_{76} - x_{16}$ ensuring the ....

2.Analyzing constraints

I'm examining the constraints of the variables, aiming to **maximize the difference** between the 76th and 16th ....

3.Setting indices

I'm setting x_{16} to **the most negative value** and x_{76} to **the most positive one**, given... .

...

6.Unraveling the equation

I'm digging into the equation $b = \frac{16}{25}a$ to....

> Inference

Step 1: Assign Extreme Values

To maximize $x_{76} - x_{16}$, we need to maximize x_{76} and minimize x_{16}.: ...

•Step 2: Apply the Sum Constraints

Next, **apply the absolute value sum constraint**: ...

Step 3: Compute $x_{76} - x_{16}$ Simplify: $\frac{82}{1600} = \frac{41}{800}$.

Step 4: Find m + n Let $m = 41, n = 800$, so $m + n = 41 + 800 = 841$.

O1's Answer: 841 ***Golden Answer: 841***

Figure 9: The o1's case of AIME.

COLLIE

>Input

Please generate a paragraph:
1) **with** exactly 3 sentences;
2) **not** containing the word 'be';
3) **not** containing the word 'of';
4) **not** containing the word 'is'.

> Thinking

****Crafting the paragraph****
I'm careful **not to use** 'be', 'of', or 'is'. Instead, I'm selecting precise, vivid words, ensuring the paragraph's structure and meaning are thoughtfully established. This approach keeps the text engaging and grammatically sound.
****Creating compliant content****
I'm crafting a **3-sentence** paragraph that **avoids** 'be', 'of', and 'is'. The paragraph will stay clear of mentioning policies or hidden reasoning, ensuring compliance and appropriateness.
****Clarifying usage rules****
I'm sorting out which forms of 'to be' are allowed, like 'being' and 'been', while 'be', 'is', 'are', 'was', and 'were' are **prohibited**. This distinction ensures precision in word choice.
****Crafting the scene****
I'm thinking about a sentence to paint a vivid picture **without** using 'of', 'be', or 'is'. The phrase "hues of pink and orange" needs rephrasing to **avoid** 'of'.
****Painting a morning****
I'm crafting a vivid morning scene: the sun rises, colors the sky in pink and orange, birds sing, and people enjoy hopeful smiles as the day starts. It's sprucing up each sentence for clarity and appeal.

>Inference

1 The sun climbs slowly, painting the horizon with vibrant shades that hint at warmth. 2 Birds sing in unison, their melody drifting across the quiet streets. 3 People stroll along paths, their faces glowing with anticipation for what lies ahead.

Grammar Check: Pass

Figure 10: The o1's case of Collie.

USACO

>Input

Given N cows (2 <= N <= 100) with a social hierarchy and specific milking position demands, find the earliest position cow 1 can be milked. The input includes:
- N: Total number of cows.
- M: Number of cows in a specific social order.
- K: Number of cows with a fixed milking position.
- A list of M integers representing the social hierarchy order.
- K pairs of integers: cow c_i and its required position p_i.

> Thinking

****Cracking the problem**** I'm analyzing the given constraints and how they shape the milking order. The objective is to find the earliest possible position for cow 1 in the milking sequence, ensuring all constraints are met.
****Mapping out constraints**** Focusing on N cows, peculiar positions for M of them, and rearrangements for K. The goal is to spot the earliest position for cow 1.
...

>Inference

To solve this problem, we need to determine the earliest position that cow 1 can occupy in the milking order while satisfying all the given constraints:

1. **Ordering Constraints**: Cows in the hierarchy must be milked in the specified order.
2. **Fixed Positions**: Certain cows must be milked at specific positions.
3. **Unique Positions**: Each cow must occupy a unique position in the order.

Approach Overview:
...
Implementation Considerations:
...

Grammar Check: Pass

Figure 11: The o1's case of USACO.