Search-o1: Agentic Search-Enhanced Large Reasoning Models

Xiaoxi Li[1], **Guanting Dong**[1], **Jiajie Jin**[1], **Yuyao Zhang**[1], **Yujia Zhou**[2], **Yutao Zhu**[1], **Peitian Zhang**[1], **Zhicheng Dou**[1*]
[1]Renmin University of China [2]Tsinghua University
{xiaoxi_li, dou}@ruc.edu.cn

Abstract

Large reasoning models (LRMs) like OpenAI-o1 have demonstrated impressive long stepwise reasoning capabilities through large-scale reinforcement learning. However, their extended reasoning processes often suffer from knowledge insufficiency, leading to frequent uncertainties and potential errors. To address this limitation, we introduce **Search-o1**, a framework that enhances LRMs with an agentic retrieval-augmented generation (RAG) mechanism and a Reason-in-Documents module for refining retrieved documents. Search-o1 integrates an agentic search workflow into the reasoning process, enabling dynamic retrieval of external knowledge when LRMs encounter uncertain knowledge points. Additionally, due to the verbose nature of retrieved documents, we design a separate Reason-in-Documents module to deeply analyze the retrieved information before injecting it into the reasoning chain, minimizing noise and preserving coherent reasoning flow. Extensive experiments on complex reasoning tasks in science, mathematics, and coding, as well as six open-domain QA benchmarks, demonstrate the strong performance of Search-o1. This approach enhances the trustworthiness of LRMs in complex reasoning tasks, paving the way for advanced deep research systems. The code is available at https://github.com/RUC-NLPIR/Search-o1.

1 Introduction

Recently emerged large reasoning models (LRMs), exemplified by OpenAI's o1 (Jaech et al., 2024), Qwen-QwQ (Team, 2024), and DeepSeek-R1 (Guo et al., 2025), employ large-scale reinforcement learning to foster impressive long-sequence stepwise reasoning capabilities, offering promising solutions to complex reasoning problems (OpenAI, 2024; Lewkowycz et al., 2022; Wei et al., 2022). This advancement has inspired a series of foundational efforts aimed at exploring and reproducing o1-like reasoning patterns, to broaden their application to a wider range of foundational models (Qin et al., 2024; Min et al., 2024).

It is noteworthy that o1-like reasoning patterns guide LRMs to engage in a slower thinking process (Daniel, 2017) by implicitly breaking down complex problems, generating a long internal reasoning chain, and then discovering suitable solutions step by step. While this characteristic enhances logical coherence and interpretability of reasoning, an extended chain of thought may cause overthinking (Chen et al., 2024b) and increased risks of knowledge insufficiency (Wong et al., 2021; Raffel et al., 2020), where any knowledge gap can propagate errors and disrupt the entire reasoning chain (Ling et al., 2023; Liu et al., 2024).

To evaluate uncertainty in LRMs' reasoning, we conducted preliminary experiments to track the frequency of uncertain words decoded by the LRMs. As shown in Figure 1, the extended thinking process leads LRM to frequently decode numerous uncertain terms in challenging reasoning problems, with "*perhaps*" averaging over 30 occurrences in each reasoning process. Consequently, automating the supplementation of knowledge required for the o1-like reasoning process has become a significant challenge, limiting the progress of LRMs in achieving universally trustworthy reasoning.

To address these challenges, we introduce **Search-o1**, which enhances LRMs by integrating an agentic retrieval-augmented generation (RAG) mechanism with a knowledge refinement module. This design allows the model to retrieve relevant knowledge as needed during the reasoning process, ensuring step-by-step reasoning remains coherent.

Specifically, our results in Figure 1 reveal that traditional problem-oriented RAG techniques do not effectively address the knowledge gaps compared to direct reasoning (Standard RAG vs. Direct

*Corresponding author.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 5421–5439
November 4-9, 2025

Cases of Model-Expressed Uncertainty
Wait, perhaps it's referring to dimethyl sulfone, but that doesn't seem right.
Alternatively, **perhaps** there's a mistake in my understanding of epistasis. Let me look up epistasis quickly. Epistasis is ...
Alternatively, HBr could also abstract a hydrogen atom from the alkene, leading to a ...
As I recall, Quinuclidine is a seven-membered ring with a nitrogen atom, **likely** not having the required symmetry.

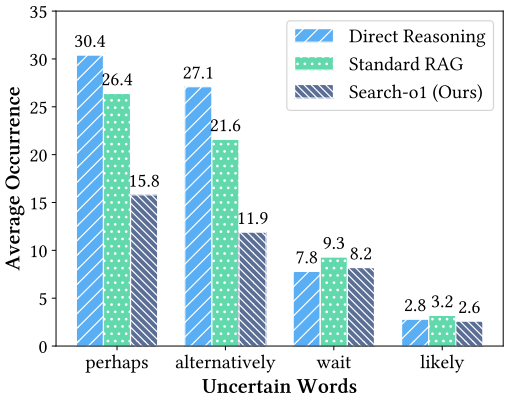

Figure 1: Analysis of reasoning uncertainty. **Left**: Examples of uncertain words identified during the reasoning process. **Right**: Average occurrence of high-frequency uncertain words per output in the GPQA diamond set.

Reasoning). This finding aligns with human intuition, as standard RAG retrieves relevant knowledge only once in a problem-oriented manner, while the knowledge required for each step in complex reasoning scenarios is often varied and diverse (Zheng et al., 2025; Liu et al., 2024; Dong et al., 2025d). Unlike them, Search-o1 employs an agentic RAG technique that guides the model to actively decode search queries when facing knowledge shortages, thereby triggering the retrieval mechanism to obtain relevant external knowledge. Owing to the benefits of this design, our retrieval mechanism can be triggered and iterated multiple times within a single reasoning session to fulfill the knowledge needs of various reasoning steps.

To effectively integrate retrieved knowledge into the LRM's reasoning process, we further identify two key challenges when directly incorporating retrieved documents into the reasoning chain during practical experiments: **(1) Redundant Information in Retrieved Documents.** Retrieved documents are often lengthy and contain redundant information, directly inputting them into LRMs may disrupt the original coherence of reasoning and even introduce noise (Wu et al., 2024; Yoran et al., 2024). **(2) Limited Ability to Understand Long Documents.** Most LRMs have been specifically aligned for complex reasoning tasks during the pre-training and fine-tuning stages. This focus has resulted in a degree of catastrophic forgetting in their general capabilities (Lin et al., 2024; Dong et al., 2024), ultimately limiting their long-context understanding of retrieved documents.

To address these challenges, we introduce the **Reason-in-Documents module**, which operates independently from the main reasoning chain. This module first conducts a thorough analysis of retrieved documents based on both the current search query and previous reasoning steps, and then produces refined information that seamlessly integrates with the prior reasoning chain.

In summary, our contributions are as follows:

- We propose Search-o1, the first framework that integrates the agentic search workflow into the o1-like reasoning process of LRM for achieving autonomous knowledge supplementation.
- To enhance reasoning with external knowledge, Search-o1 incorporates an agentic RAG mechanism and a knowledge refinement module. This enables LRMs to retrieve relevant external information as needed, while preserving the coherence of the reasoning process.
- With five complex reasoning domains and six open-domain QA benchmarks, we demonstrate that Search-o1 achieves remarkable performance in the reasoning field while maintaining substantial improvements in the general knowledge.

2 Related Work

Large Reasoning Models. Large reasoning models focus on enhancing performance at test time by utilizing extended reasoning steps, contrasting with traditional large pre-trained models that achieve scalability during training by increasing model size or expanding training data (Henighan et al., 2020; Zeng et al., 2024). Recently, models like OpenAI-o1 (Jaech et al., 2024), Qwen-QwQ (Team, 2024) and DeepSeek-R1 (Guo et al., 2025) explicitly demonstrate chain-of-thought reasoning (Wei et al., 2022). Various approaches have been explored to achieve o1-like reasoning capabilities. Some methods combine policy and reward models with Monte

Carlo Tree Search (MCTS) (Jiang et al., 2024), though this does not internalize reasoning within the model. Other studies incorporate deliberate errors in reasoning paths during training to partially internalize these abilities (Qin et al., 2024). Additionally, distilling training data has been shown to enhance models' o1-like reasoning skills (Min et al., 2024). The o1-like reasoning paradigm has demonstrated strong performance across diverse domains, including vision-language reasoning (Xu et al., 2024b; Dong et al., 2025d), code generation (Zhang et al., 2024), healthcare (Chen et al., 2024a), and machine translation (Wang et al., 2024). However, these approaches are limited by their reliance on static, parameterized models, which cannot leverage external world knowledge when internal knowledge is insufficient.

Retrieval-Augmented Generation. Retrieval-augmented generation (RAG) enhances generative models by incorporating retrieval mechanisms, overcoming the limitations of static parameters and enabling access to external knowledge (Lewis et al., 2020; Li et al., 2025c). Advanced research in this field enhances the RAG system from multiple aspects, including the necessity of retrieval (Tan et al., 2024), pre-processing of queries (Wang et al., 2023), retrieved documents compressing (Xu et al., 2024a; Liu et al., 2025), denoising (Liu et al., 2024), refining (Jin et al., 2025a), instruction following (Dong et al., 2025b), and so on. Furthermore, some studies have explored end-to-end model training to implement RAG systems (Li et al., 2024, 2025b) and knowledge-graph-based RAG systems (Edge et al., 2024). Recently, agentic RAG systems empower models to autonomously determine when and what knowledge to retrieve as needed, showcasing enhanced planning and problem-solving capabilities (Yao et al., 2023; Dong et al., 2025a; Li et al., 2025a; Jin et al., 2025b; Dong et al., 2025c). However, existing RAG approaches have not incorporated the strong reasoning capabilities of o1-like models, limiting the potential to further enhance system performance in solving complex tasks.

3 Methodology

3.1 Problem Formulation

We consider a complex reasoning task that necessitates multi-step reasoning and the retrieval of external knowledge to derive solutions. The objective is to generate a comprehensive solution for each question q, consisting of both a logical reasoning chain $\mathcal{R}$ and the final answer a. In this work, we enable the reasoning model to utilize external knowledge sources during the reasoning process. Specifically, we consider three primary inputs in the problem-solving process: the task instruction I, the question q, and externally retrieved documents $\mathcal{D}$. Here, I provides an overarching description of the reasoning task, q is the specific complex question to be answered, and $\mathcal{D}$ comprises background knowledge dynamically retrieved from an external corpus.

The goal is to design a reasoning mechanism that effectively integrates I, q, and $\mathcal{D}$ to produce a coherent reasoning chain $\mathcal{R}$ and a final answer a. This can be formalized as the mapping $(I, q, \mathcal{D}) \rightarrow (\mathcal{R}, a)$. The generation of the reasoning sequence and the final answer can be expressed as:

$$\begin{aligned} &P(\mathcal{R}, a \mid I, q, \mathcal{D}) = \\ &\underbrace{\prod_{t=1}^{T_r} P(\mathcal{R}_t \mid \mathcal{R}_{<t}, I, q, \mathcal{D}_{<t})}_{\text{Reasoning Process}} \cdot \\ &\underbrace{\prod_{t=1}^{T_a} P(a_t \mid a_{<t}, \mathcal{R}, I, q)}_{\text{Answer Generation}}, \end{aligned} \tag{1}$$

where T_r is the number of tokens in the reasoning sequence $\mathcal{R}$. The token at the position t is $\mathcal{R}_t$, and $\mathcal{R}_{<t}$ represents all tokens generated before position t. $\mathcal{D}_{\leq t}$ represents all documents retrieved up to token t in the reasoning chain. Similarly, T_a is the length of the answer sequence a, with a_t being the token at the position t and $a_{<t}$ indicating all generated answer tokens before the position t.

3.2 Overview of the Search-o1 Framework

The Search-o1 framework addresses knowledge insufficiency in large reasoning models (LRMs) by seamlessly integrating external knowledge retrieval into their reasoning process while maintaining chain-of-thought coherence. As illustrated in Figure 2, we present a comparative analysis of three approaches: vanilla reasoning, agentic retrieval-augmented generation (RAG), and our proposed Search-o1 framework.

Vanilla Reasoning Pattern. Consider the example in Figure 2(a), where the task involves determining the carbon atom count in the final product of a three-step chemical reaction. The vanilla reasoning approach falters when encountering knowledge gaps (e.g., the "structure of trans-Cinnamaldehyde"). Without access to accurate

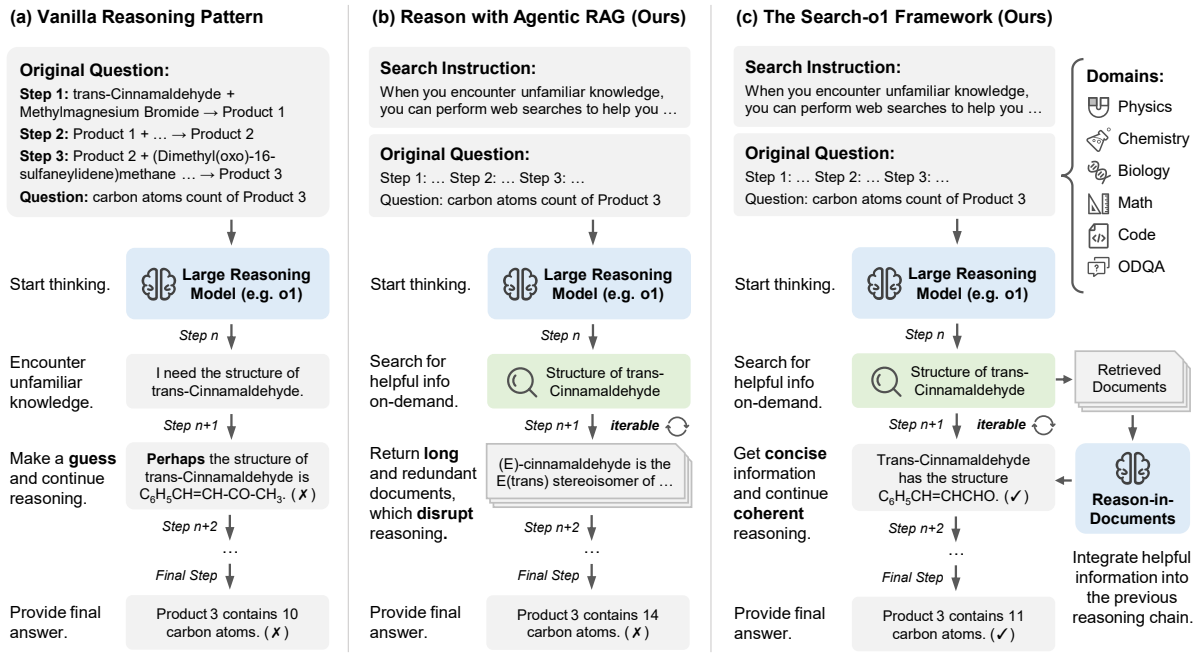

Figure 2: Comparison of reasoning approaches: (a) Direct reasoning without retrieval often results in inaccuracies due to missing knowledge. (b) Our agentic retrieval-augmented reasoning approach improves knowledge access but usually returns lengthy, redundant documents, disrupting coherent reasoning. (c) Our Search-o1 integrates concise retrieved knowledge seamlessly into the reasoning process, enabling precise and coherent problem-solving.

information, the model must rely on assumptions, potentially leading to cascading errors throughout subsequent reasoning steps.

Agentic RAG. To bridge the knowledge gaps during reasoning, we build the agentic RAG mechanism (Figure 2(b)) to enable the model to autonomously retrieve external knowledge when needed. When uncertainty arises—such as regarding the compound's structure—the model generates targeted search queries (e.g., "structure of trans-Cinnamaldehyde"). However, the direct insertion of retrieved documents, which often contain lengthy and irrelevant information, may disrupt the reasoning flow and hurt coherence.

Search-o1. Our Search-o1 framework (Figure 2(c)) extends the agentic RAG mechanism by incorporating a Reason-in-Documents module. This module condenses retrieved documents into focused reasoning steps that integrate external knowledge while maintaining the logical flow of the reasoning chain. It considers the current search query, retrieved documents, and the existing reasoning chain to generate coherent steps. This iterative process continues until the final answer is reached.

3.3 Agentic Retrieval-Augmented Generation Mechanism

The agentic RAG mechanism is a pivotal component of the Search-o1 framework, empowering the reasoning model to autonomously determine when to retrieve external knowledge during the reasoning process. This mechanism allows the model itself to decide whether to continue generating reasoning steps or to initiate a retrieval step. Detailed model instructions can be found in Appendix A.1.

During the generation of the reasoning chain $\mathcal{R}$, the model may intermittently generate search queries $q_{\text{search}}^{(i)}$ encapsulated between special symbols <|begin_search_query|> and <|end_search_query|>, where i indexes the i-th search step. Each search query is generated based on the current state of the reasoning process and the previously retrieved knowledge. The generation of each search query is expressed as: $P(q_{\text{search}}^{(i)} \mid I, q, \mathcal{R}^{(i-1)}) = \prod_{t=1}^{T_q^{(i)}} P\left(q_{\text{search},t}^{(i)} \mid q_{\text{search},<t}^{(i)}, I, q, \mathcal{R}^{(i-1)}\right)$, where $T_q^{(i)}$ is the length of the i-th search query, $q_{\text{search},t}^{(i)}$ denotes the token generated at step t of the i-th search query, and $\mathcal{R}^{(i-1)}$ represents all the reasoning steps prior to the i-th search step, including both search queries and search results.

Once a new pair of special symbols for the search

query is detected in the reasoning sequence, we pause the reasoning process, and the search query $q_{\text{search}}^{(i)}$ is extracted. The retrieval function `Search` is invoked to obtain relevant documents:

$$\mathcal{D}^{(i)} = \texttt{Search}(q_{\text{search}}^{(i)}), \tag{2}$$

where $\mathcal{D}^{(i)} = d_1^{(i)}, d_2^{(i)}, \ldots, d_{k_i}^{(i)}$ represents the set of top-k_i relevant documents retrieved for the i-th search query. The retrieved documents $\mathcal{D}^{(i)}$ are subsequently injected into the reasoning chain $\mathcal{R}^{(i-1)}$ between the special symbols <|begin_search_result|> and <|end_search_result|>, allowing the reasoning model to utilize the external knowledge to continue the reasoning process.

This agentic mechanism enables the model to dynamically and efficiently incorporate external knowledge, maintaining the coherence and relevance of the reasoning process while avoiding information overload from excessive or irrelevant retrieval results.

3.4 Knowledge Refinement via Reason-in-Documents

While the agentic RAG mechanism addresses knowledge gaps in reasoning, directly inserting full documents can disrupt coherence due to their length and redundancy. To overcome this, the Search-o1 framework includes the knowledge refinement module, which selectively integrates only relevant and concise information into the reasoning chain through a separate generation process using the original reasoning model. This module processes retrieved documents to align with the model's specific reasoning needs, transforming raw information into refined, pertinent knowledge while maintaining coherence and logical consistency of the main reasoning chain.

The refinement guidelines for Reason-in-Documents are detailed in Appendix A.1. These guidelines instruct the model to analyze the retrieved web pages based on the previous reasoning steps, current search query, and the content of the searched web pages. The objective is to extract relevant and accurate information that directly contributes to advancing the reasoning process for the original question, ensuring seamless integration into the existing reasoning chain.

Formally, for each search step i, let $\mathcal{R}^{(<i)}$ denote the reasoning chain accumulated up to just before the i-th search query. Given $\mathcal{R}^{(<i)}$, the current search query $q_{\text{search}}^{(i)}$, and the retrieved documents $\mathcal{D}^{(i)}$, the knowledge refinement process operates in two stages: first generating an intermediate reasoning sequence $r_{\text{docs}}^{(i)}$ to analyze the retrieved documents, then producing refined knowledge $r_{\text{final}}^{(i)}$ based on this analysis. The generation of the intermediate reasoning sequence $r_{\text{docs}}^{(i)}$ is expressed as: $P(r_{\text{docs}}^{(i)} \mid \mathcal{R}^{(<i)}, q_{\text{search}}^{(i)}, \mathcal{D}^{(i)}) = \prod_{t=1}^{T_d^{(i)}} P\left(r_{\text{docs},t}^{(i)} \mid r_{\text{docs},<t}^{(i)}, \mathcal{R}^{(<i)}, q_{\text{search}}^{(i)}, \mathcal{D}^{(i)}\right)$, where $T_d^{(i)}$ is the length of the intermediate reasoning sequence, and $r_{\text{docs},t}^{(i)}$ denotes the token at step t. The refined knowledge $r_{\text{final}}^{(i)}$ is then generated based on this analysis: $P(r_{\text{final}}^{(i)} \mid r_{\text{docs}}^{(i)}, \mathcal{R}^{(<i)}, q_{\text{search}}^{(i)}) = \prod_{t=1}^{T_r^{(i)}} P\left(r_{\text{final},t}^{(i)} \mid r_{\text{final},<t}^{(i)}, r_{\text{docs}}^{(i)}, \mathcal{R}^{(<i)}, q_{\text{search}}^{(i)}\right)$, where $T_r^{(i)}$ is the length of the refined knowledge sequence, and $r_{\text{final},t}^{(i)}$ denotes the token at step t.

The refined knowledge $r_{\text{final}}^{(i)}$ is then incorporated into the reasoning chain $\mathcal{R}^{(i)}$, enabling the model to continue generating coherent reasoning steps with access to the external knowledge.

$$\begin{aligned} &P(\mathcal{R}, a \mid I, q) = \\ &\prod_{t=1}^{T_r} P\left(\mathcal{R}_t \mid \mathcal{R}_{<t}, I, q, \{r_{\text{final}}^{(j)}\}_{j \le i(t)}\right) \cdot \\ &\prod_{t=1}^{T_a} P\left(a_t \mid a_{<t}, \mathcal{R}, I, q\right), \end{aligned} \tag{3}$$

where $\{r_{\text{final}}^{(j)}\}_{j \le i(t)}$ denotes all previously refined knowledge up to the $i(t)$-th search step. Here, $i(t)$ represents the index of the search step corresponding to the current reasoning step t. This refined knowledge integration ensures that each reasoning step can access relevant external information while maintaining the conciseness and focus of the reasoning process.

3.5 Search-o1 Inference Process

The Search-o1 inference process starts by combining the task instruction I with the specific question q. As the reasoning model $\mathcal{M}$ generates the reasoning chain $\mathcal{R}$, it may create search queries marked by <|begin_search_query|> and <|end_search_query|>. When the <|end_search_query|> symbol is detected, the search query q_{search} is extracted and used to fetch relevant documents $\mathcal{D}$ through the `Search` function. These documents, along with the instruction I_{docs} and the reasoning chain $\mathcal{R}$, are processed by the Reason-in-Documents module, which refines the information into a more concise and relevant form r_{final}. This refined information is then added back into

Table 1: Main results on challenging reasoning tasks, including PhD-level science QA, math, and code benchmarks. We report Pass@1 metric for all tasks. For models with 32B parameters, the best results are in **bold** and the second-best are underlined. Results from larger or non-proprietary models are in gray color for reference. Symbol "†" indicates results from their official releases.

Method	GPQA (PhD-Level Science QA)				Math Benchmarks			LiveCodeBench			
	Physics	Chemistry	Biology	Overall	MATH500	AMC23	AIME24	Easy	Medium	Hard	Overall
Direct Reasoning (w/o Retrieval)											
Qwen2.5-32B	57.0	33.3	52.6	45.5	75.8	57.5	23.3	42.3	18.9	14.3	22.3
Qwen2.5-Coder-32B	37.2	25.8	57.9	33.8	71.2	67.5	20.0	61.5	16.2	12.2	25.0
QwQ-32B	75.6	39.8	68.4	58.1	83.2	82.5	53.3	61.5	29.7	**20.4**	**33.0**
Qwen2.5-72B	57.0	37.6	68.4	49.0	79.4	67.5	20.0	53.8	29.7	24.5	33.0
Llama3.3-70B	54.7	31.2	52.6	43.4	70.8	47.5	36.7	57.7	32.4	24.5	34.8
DeepSeek-R1-Lite†	-	-	-	58.5	91.6	-	52.5	-	-	-	51.6
GPT-4o†	59.5	40.2	61.6	50.6	60.3	-	9.3	-	-	-	33.4
o1-preview†	89.4	59.9	65.9	73.3	85.5	-	44.6	-	-	-	53.6
Retrieval-augmented Reasoning											
RAG-Qwen2.5-32B	57.0	37.6	52.6	47.5	82.6	72.5	30.0	61.5	24.3	8.2	25.9
RAG-QwQ-32B	76.7	38.7	73.7	58.6	84.8	82.5	50.0	57.7	16.2	12.2	24.1
RAgent-Qwen2.5-32B	58.1	33.3	63.2	47.0	74.8	65.0	20.0	57.7	24.3	6.1	24.1
RAgent-QwQ-32B	76.7	46.2	68.4	61.6	85.0	**85.0**	**56.7**	**65.4**	18.9	12.2	26.8
Retrieval-augmented Reasoning with Reason-in-Documents											
Search-o1 (Ours)	**77.9**	**47.3**	**78.9**	**63.6**	**86.4**	**85.0**	**56.7**	57.7	**32.4**	**20.4**	**33.0**

the reasoning chain $\mathcal{R}$ within <|begin_search_result|> and <|end_search_result|> symbols. The process repeats to ensure the reasoning model includes external knowledge while keeping the reasoning coherent and logically consistent, leading to the final answer a. For more details, refer to Appendix B.

4 Experiments

4.1 Tasks and Datasets

Challenging reasoning tasks: (1) **GPQA** (Rein et al., 2023) is a PhD-level science multiple-choice QA dataset. The questions are authored by domain experts in physics, chemistry, and biology. (2) **Math benchmarks** include **MATH500** (Lightman et al., 2024), **AMC2023**, and **AIME2024**. MATH500 consists of 500 questions from the MATH test set (Hendrycks et al., 2021). AMC2023 and AIME2024 are middle school math competitions covering arithmetic, algebra, geometry, etc., containing 40 and 30 questions respectively. (3) **LiveCodeBench** (Jain et al., 2024) is a benchmark for evaluating LLMs' coding capabilities, consisting of easy, medium, and hard difficulty problems.

Open-domain QA tasks: (1) **Single-hop QA datasets: Natural Questions (NQ)** (Kwiatkowski et al., 2019) contains questions from Google search queries with answers from Wikipedia. **TriviaQA** (Joshi et al., 2017) includes questions from trivia websites and competitions. (2) **Multi-hop QA datasets: HotpotQA** (Yang et al., 2018) requires reasoning across multiple Wikipedia paragraphs. **2WikiMultihopQA (2WIKI)** (Ho et al., 2020) provides explicit reasoning paths. **MuSiQue** (Trivedi et al., 2022) consists of 2-4 hop questions derived from single-hop datasets. **Bamboogle** (Press et al., 2023) includes complex questions that Google answers incorrectly, assessing compositional reasoning.

4.2 Baselines

Direct Reasoning. These methods use the model's internal knowledge without retrieval. Open-source models include Qwen2.5-32B-Instruct (Qwen et al., 2024), Qwen2.5-Coder-32B-Instruct (Hui et al., 2024), QwQ-32B-Preview (Team, 2024), Qwen2.5-72B-Instruct (Qwen et al., 2024), Llama3.3-70B-Instruct (Dubey et al., 2024), and DeepSeek-R1-Lite (Guo et al., 2025). Closed-source models include OpenAI GPT-4o (Hurst et al., 2024) and o1-preview (Jaech et al., 2024).

Retrieval-augmented Reasoning. These methods use external information for reasoning. We consider: **(1) Standard RAG:** Retrieves the top-10 documents for reasoning and answer generation. **(2) RAG Agent (RAgent):** Lets the model decide when to query for retrieval, as described in Sec-

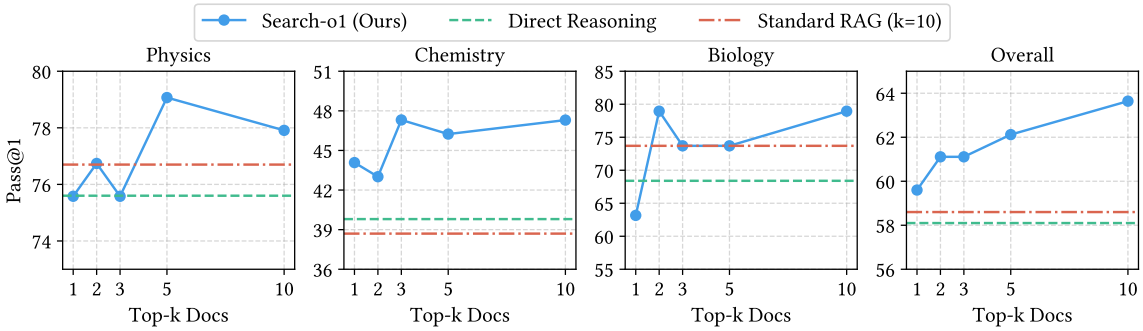

Figure 3: Scaling analysis of top-k retrieved documents utilized in reasoning.

Table 2: Performance comparison with different backbone large reasoning models.

Method	GPQA	MATH500	AIME24	LiveCode
Sky-T1-32B-Preview				
Direct Reasoning	57.6	83.2	43.3	26.8
RAG-Agent	59.1	83.8	46.7	25.0
Search-o1	**60.6**	**84.4**	**50.0**	**27.7**
DeepSeek-R1-Distill-Qwen-32B				
Direct Reasoning	65.7	88.2	**73.3**	42.9
RAG-Agent	66.2	88.4	70.0	42.0
Search-o1	**67.2**	**89.0**	**73.3**	**43.8**

Table 3: Performance comparison with human experts on the GPQA extended set (Rein et al., 2023).

Method	Physics	Chemistry	Biology	Overall
Human Experts				
Physicists	57.9	31.6	42.0	39.9
Chemists	34.5	**72.6**	45.6	48.9
Biologists	30.4	28.8	68.9	37.2
Reasoning Models				
QwQ-32B	61.7	36.9	61.0	51.8
RAG-QwQ-32B	64.3	38.3	66.7	54.6
Search-o1	**68.7**	40.7	**69.5**	**57.9**

tion 3.3. Inspired by ReAct (Yao et al., 2023), we first retrieve the top-10 snippets and then fetch full documents when needed.

4.3 Implementation Details

For the backbone large reasoning model in Search-o1, we utilize the open-sourced QwQ-32B-Preview. For generation settings, we use a maximum of 32,768 tokens, temperature of 0.7, top_p of 0.8, top_k of 20, and a repetition penalty of 1.05 across all models. For retrieval, we employ the Bing Web Search API, setting the region to US-EN and the top-k retrieved documents to 10. For baseline models not specifically trained for o1-like reasoning, we apply Chain-of-Thought prompting to perform reasoning before generating answers. Detailed instructions for all models are provided in Appendix A.

4.4 Results on Challenging Reasoning Tasks

Main Results. Table 1 shows Search-o1's performance on complex reasoning tasks: (1) Reasoning model QwQ-32B-Preview consistently outperforms traditional general LLMs in both retrieval and non-retrieval settings, even surpassing larger models like Qwen2.5-72B and Llama3.3-70B in direct reasoning, highlighting the effectiveness of the o1-like long CoT approach. **(2) RAgent-QwQ-32B outperforms both standard RAG models and QwQ-32B in most tasks**, thanks to its agentic search mechanism, which autonomously retrieves information to supplement knowledge required for reasoning at each step. In contrast, the non-reasoning model Qwen2.5-32B with agentic RAG performs similarly to standard RAG on GPQA but worse on math and code tasks, indicating that general LLMs struggle with complex reasoning. **(3) Search-o1 surpasses RAgent-QwQ-32B in most tasks,** demonstrating the effectiveness of the Reason-in-Documents strategy. Specifically, on average across all five datasets, Search-o1 exceeds RAgent-QwQ-32B and QwQ-32B by 4.7% and 3.1%, respectively, and significantly outperforms non-reasoning LLMs Qwen2.5-32B and Llama3.3-70B by 44.7% and 39.3%.

Scaling Analysis on Number of Retrieved Documents. We analyze performance variation with respect to the number of retrieved documents (Figure 3). **Search-o1 effectively leverages an increasing number of retrieved documents, improving its handling of complex reasoning tasks**. Notably, retrieving even one document can outperform Direct Reasoning and standard RAG models using ten documents, showcasing the power of agentic search and Reason-in-Documents strategies.

Table 4: Performance comparison on open-domain QA tasks, including single-hop QA and multi-hop QA datasets. For models with 32B parameters, the best results are in **bold** and the second-best are underlined. Results from larger models are in gray color for reference.

Method	Single-hop QA				Multi-hop QA							
	NQ		TriviaQA		HotpotQA		2WIKI		MuSiQue		Bamboogle	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Direct Reasoning (w/o Retrieval)												
Qwen2.5-32B	22.8	33.9	52.0	60.3	25.4	34.7	29.8	36.3	8.4	18.0	49.6	63.2
QwQ-32B	23.0	33.1	53.8	60.7	25.4	33.3	34.4	40.9	9.0	18.9	38.4	53.7
Qwen2.5-72B	27.6	41.2	56.8	65.8	29.2	38.8	34.4	42.7	11.4	20.4	47.2	61.7
Llama3.3-70B	36.0	48.7	68.8	76.8	37.8	49.1	46.0	54.2	14.8	23.6	54.4	67.8
Retrieval-augmented Reasoning												
RAG-Qwen2.5-32B	33.4	49.3	**65.8**	**79.2**	38.6	50.4	31.6	40.6	10.4	19.8	52.0	66.0
RAG-QwQ-32B	29.6	44.4	65.6	77.6	34.2	46.4	35.6	46.2	10.6	20.2	55.2	67.4
RAgent-Qwen2.5-32B	32.4	47.8	63.0	72.6	44.6	56.8	55.4	69.7	13.0	25.4	54.4	66.4
RAgent-QwQ-32B	33.6	48.4	62.0	74.0	43.0	55.2	**58.4**	71.2	13.6	25.5	52.0	64.7
Retrieval-augmented Reasoning with Reason-in-Documents												
Search-o1 (Ours)	**34.0**	**49.7**	63.4	74.1	**45.2**	**57.3**	58.0	**71.4**	**16.6**	**28.2**	**56.0**	**67.8**

Analysis of Performance with Different Backbone LRMs. We examine the effectiveness of integrating search with different reasoning models, specifically Sky-T1-32B-Preview and DeepSeek-R1-Distill-Qwen-32B (Figure 2). The results show that **RAgent consistently outperforms direct reasoning, with Search-o1 delivering the highest performance across all tasks**. These findings underscore the robustness of the Search-o1 framework, which enhances the reasoning capabilities of various LRMs and lays a solid foundation for real-world applications.

Comparison with Human Experts. We compare Search-o1's performance with human experts across various domains in the GPQA extended set (Table 3). Search-o1 outperforms human experts overall (57.9), particularly in physics (68.7) and biology (69.5). Although it trails chemists in chemistry (40.7 vs. 72.6), it remains competitive overall, highlighting **Search-o1's ability to leverage document retrieval and reasoning to achieve expert-level cross-domain performance.**

4.5 Results on Open-Domain QA Tasks

In addition to the reasoning tasks where LRMs excel, we also explore the performance of our Search-o1 on open-domain QA tasks. Table 4 shows the results of Search-o1 on open-domain QA tasks: (1) Without retrieval, QwQ-32B performs similarly to Qwen2.5-32B, with slight performance drop (31.3 vs. 30.7 EM), suggesting **LRMs are weaker on open-domain QA tasks than on reasoning tasks.** (2) Retrieval significantly improves performance for both reasoning and non-reasoning models, indicating knowledge gaps in these tasks. Agentic RAG boosts QwQ-32B's performance by 23.2% on multi-hop QA tasks, but shows little change in single-hop tasks (47.8 vs. 47.6 EM), proving **agentic search can better unleash the potential of LRMs in more complex QA tasks**. (3) **For our Search-o1, we find that it generally outperforms all baselines on multi-hop tasks.** Specifically, in terms of the average EM metric, our Search-o1 exceeds RAG-QwQ-32B and RAgent-QwQ-32B by 29.6% and 5.3%, respectively, demonstrating the effectiveness of our Reason-in-Documents strategy in complex QA tasks. **This further emphasizes the importance of maintaining consistency between external knowledge and the logical chain of reasoning.**

5 Conclusion

In this work, we present Search-o1, a framework that addresses the knowledge insufficiency inherent in large reasoning models (LRMs) by integrating an agentic retrieval-augmented generation mechanism alongside a Reason-in-Documents module. Our approach enables LRMs to autonomously retrieve and seamlessly incorporate external knowledge during the reasoning process, thereby enhancing both the accuracy and coherence of their long-step reasoning capabilities. Comprehensive experiments across diverse complex reasoning tasks in science, mathematics, and coding, as well as mul-

tiple open-domain QA benchmarks, demonstrate that Search-o1 consistently outperforms existing retrieval-augmented and direct reasoning methods. This work opens up the potential of integrating tools such as search into LRMs, paving the way for more trustworthy and effective intelligent systems in complex problem-solving scenarios.

6 Limitations

Search-o1 has three main limitations that future research could address: (1) it currently enhances o1-like reasoning models solely through search tools. Future work could incorporate additional tools, such as calculators, code interpreters, and various real-world APIs; (2) this work focuses on using instructions to enable reasoning models to leverage search capabilities. To improve search execution and the use of search results, future research could explore fine-tuning and reinforcement learning techniques to further optimize Search-o1; and (3) this study focuses on large reasoning language models. Expanding the Search-o1 framework to support multimodal reasoning models represents a promising avenue for future exploration.

Acknowledgment

This work was supported by Beijing Municipal Science and Technology Project No. Z231100010323009, National Natural Science Foundation of China No. 62272467, Beijing Natural Science Foundation No. L233008. The work was partially done at the Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE.

References

Junying Chen, Zhenyang Cai, Ke Ji, Xidong Wang, Wanlong Liu, Rongsheng Wang, Jianye Hou, and Benyou Wang. 2024a. Huatuogpt-o1, towards medical complex reasoning with llms. *arXiv preprint arXiv:2412.18925*.

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

Kahneman Daniel. 2017. *Thinking, fast and slow*.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. 2025a. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *CoRR*, abs/2505.16410.

Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. 2025b. Self-play with execution feedback: Improving instruction-following capabilities of large language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. 2025c. Agentic reinforced policy optimization. *Preprint*, arXiv:2507.19849.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. How abilities in large language models are affected by supervised fine-tuning data composition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 177–198. Association for Computational Linguistics.

Guanting Dong, Chenghao Zhang, Mengjie Deng, Yutao Zhu, Zhicheng Dou, and Ji-Rong Wen. 2025d. Progressive multimodal reasoning via active retrieval. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 3579–3602. Association for Computational Linguistics.

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*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *Preprint*, arXiv:2404.16130.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

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*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, An Yang, Rui Men, Fei Huang, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. 2024. Qwen2.5-coder technical report. *CoRR*, abs/2409.12186.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

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. *CoRR*, abs/2403.07974.

Jinhao Jiang, Zhipeng Chen, Yingqian Min, Jie Chen, Xiaoxue Cheng, Jiapeng Wang, Yiru Tang, Haoxiang Sun, Jia Deng, Wayne Xin Zhao, et al. 2024. Technical report: Enhancing llm reasoning with reward-guided tree search. *arXiv preprint arXiv:2411.11694*.

Jiajie Jin, Xiaoxi Li, Guanting Dong, Yuyao Zhang, Yutao Zhu, Yongkang Wu, Zhonghua Li, Ye Qi, and Zhicheng Dou. 2025a. Hierarchical document refinement for long-context retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 3502–3520. Association for Computational Linguistics.

Jiajie Jin, Xiaoxi Li, Guanting Dong, Yuyao Zhang, Yutao Zhu, Zhao Yang, Hongjin Qian, and Zhicheng Dou. 2025b. Decoupled planning and execution: A hierarchical reasoning framework for deep search. *CoRR*, abs/2507.02652.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Xiaoxi Li, Zhicheng Dou, Yujia Zhou, and Fangchao Liu. 2024. Corpuslm: Towards a unified language model on corpus for knowledge-intensive tasks. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 26–37. ACM.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. 2025a. Webthinker: Empowering large reasoning models with deep research capability. *CoRR*, abs/2504.21776.

Xiaoxi Li, Jiajie Jin, Yujia Zhou, Yongkang Wu, Zhonghua Li, Ye Qi, and Zhicheng Dou. 2025b. Retrollm: Empowering large language models to retrieve fine-grained evidence within generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 16754–16779. Association for Computational Linguistics.

Xiaoxi Li, Jiajie Jin, Yujia Zhou, Yuyao Zhang, Peitian Zhang, Yutao Zhu, and Zhicheng Dou. 2025c. From matching to generation: A survey on generative information retrieval. *ACM Trans. Inf. Syst.*, 43(3):83:1–83:62.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations,*

ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net.

Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Raghavi Chandu, Chandra Bhagavatula, and Yejin Choi. 2024. The unlocking spell on base llms: Rethinking alignment via in-context learning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Jingyu Liu, Jiaen Lin, and Yong Liu. 2024. How much can RAG help the reasoning of llm? *CoRR*, abs/2410.02338.

Wenhan Liu, Xinyu Ma, Weiwei Sun, Yutao Zhu, Yuchen Li, Dawei Yin, and Zhicheng Dou. 2025. Reasonrank: Empowering passage ranking with strong reasoning ability. *Preprint*, arXiv:2508.07050.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, et al. 2024. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *arXiv preprint arXiv:2412.09413*.

OpenAI. 2024. Learning to reason with llms.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 5687–5711. Association for Computational Linguistics.

Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, et al. 2024. O1 replication journey: A strategic progress report–part 1. *arXiv preprint arXiv:2410.18982*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022.

Jiejun Tan, Zhicheng Dou, Yutao Zhu, Peidong Guo, Kun Fang, and Ji-Rong Wen. 2024. Small models, big insights: Leveraging slim proxy models to decide when and what to retrieve for llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 4420–4436. Association for Computational Linguistics.

Qwen Team. 2024. Qwq: Reflect deeply on the boundaries of the unknown.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Jiaan Wang, Fandong Meng, Yunlong Liang, and Jie Zhou. 2024. Drt-o1: Optimized deep reasoning translation via long chain-of-thought. *arXiv preprint arXiv:2412.17498*.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9414–9423. Association for Computational Linguistics.

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

Ken C. L. Wong, Hongzhi Wang, Etienne E. Vos, Bianca Zadrozny, Campbell D. Watson, and Tanveer F. Syeda-Mahmood. 2021. Addressing deep learning model uncertainty in long-range climate forecasting with late fusion. *CoRR*, abs/2112.05254.

Siye Wu, Jian Xie, Jiangjie Chen, Tinghui Zhu, Kai Zhang, and Yanghua Xiao. 2024. How easily do irrelevant inputs skew the responses of large language models? *Preprint*, arXiv:2404.03302.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024a. RECOMP: improving retrieval-augmented lms with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Guowei Xu, Peng Jin, Li Hao, Yibing Song, Lichao Sun, and Li Yuan. 2024b. Llava-o1: Let vision language models reason step-by-step. *arXiv preprint arXiv:2411.10440*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2024. Making retrieval-augmented language models robust to irrelevant context. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Bo Wang, Shimin Li, Yunhua Zhou, Qipeng Guo, Xuanjing Huang, and Xipeng Qiu. 2024. Scaling of search and learning: A roadmap to reproduce o1 from reinforcement learning perspective. *arXiv preprint arXiv:2412.14135*.

Yuxiang Zhang, Shangxi Wu, Yuqi Yang, Jiangming Shu, Jinlin Xiao, Chao Kong, and Jitao Sang. 2024. o1-coder: an o1 replication for coding. *arXiv preprint arXiv:2412.00154*.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 1009–1024. Association for Computational Linguistics.

Appendix

A Instruction Templates

This section outlines the instructions utilized in this work, including those for the Search-o1 main reasoning chain, Reason-in-Documents, standard RAG, RAG agent, and task-specific guidelines.

A.1 Instructions for Search-o1

Instruction for Search-o1

You are a reasoning assistant with the ability to perform web searches to help you answer the user's question accurately. You have special tools:
To perform a search: write <|begin_search_query|> your query here <|end_search_query|>.
Then, the system will search and analyze relevant web pages, then provide you with helpful information in the format <|begin_search_result|> ...search results... <|end_search_result|>.
You can repeat the search process multiple times if necessary. The maximum number of search attempts is limited to {MAX_SEARCH_LIMIT}.
Once you have all the information you need, continue your reasoning.
Example:
Question: "..."
Assistant thinking steps:
- I might need to look up details about ...
Assistant:
<|begin_search_query|>...<|end_search_query|>
(System returns processed information from relevant web pages)
Assistant continues reasoning with the new information...
Remember:
- Use <|begin_search_query|> to request a web search and end with <|end_search_query|>.
- When done searching, continue your reasoning.

Instruction for Reason-in-Documents

Task Instruction:
You are tasked with reading and analyzing web pages based on the following inputs: Previous Reasoning Steps, Current Search Query, and Searched Web Pages. Your objective is to extract relevant and helpful information for Current Search Query from the Searched Web Pages and seamlessly integrate this information into the Previous Reasoning Steps to continue reasoning for the original question.
Guidelines:
1. Analyze the Searched Web Pages:
- Carefully review the content of each searched web page.
- Identify factual information that is relevant to the Current Search Query and can aid in the reasoning process for the original question.
2. Extract Relevant Information:
- Select the information from the Searched Web Pages that directly contributes to advancing the Previous Reasoning Steps.
- Ensure that the extracted information is accurate and relevant.
3. Output Format:
- If the web pages provide helpful information for current search query: Present the information beginning with 'Final Information' as shown below.
Final Information
[Helpful information]
- If the web pages do not provide any helpful information for current search query: Output the following text.
Final Information
No helpful information found.
Inputs:
- Previous Reasoning Steps:
{prev_reasoning}
- Current Search Query:
{search_query}
- Searched Web Pages:
{document}
Now you should analyze each web page and find helpful information based on the current search query "{search_query}" and previous reasoning steps.

A.2 Instructions for Standard RAG

Instruction for Standard RAG

You are a knowledgeable assistant that utilizes the provided documents to answer the user's question accurately.
Question: {question}
Documents: {documents}
Guidelines:
- Analyze the provided documents to extract relevant information. Synthesize the information to formulate a coherent and accurate answer.
- Ensure that your response directly addresses the user's question using the information from the documents.

A.3 Instructions for RAG Agent

Instruction for RAG Agent

You are a reasoning assistant with the ability to perform web searches and retrieve webpage content to help you answer the user's question accurately. You have special tools:
- To perform a search: Write '<|begin_search_query|>' your query here '<|end_search_query|>'.
The system will call the web search API with your query and return the search results in the format '<|begin_search_result|> ...search results... <|end_search_result|>'.
The search results will include a list of webpages with titles, URLs, and snippets (but not full content).
- To retrieve full page content: After receiving the search results, if you need more detailed information from specific URLs, write '<|begin_url|> url1, url2, ... <|end_url|>'.
The system will fetch the full page content of those URLs and return it as '<|begin_full_page|> ...full page content... <|end_full_page|>'.
You can repeat the search process multiple times if necessary. The maximum number of search attempts is limited to {MAX_SEARCH_LIMIT}. You can fetch up to {MAX_URL_FETCH} URLs for detailed information.
Once you have all the information you need, continue your reasoning.
Example:
Question: "..."
Assistant thinking steps: - I need to find out ...
Assistant: '<|begin_search_query|>...<|end_search_query|>'
(System returns search results)
Assistant: '<|begin_search_result|> ...search results without full page... <|end_search_result|>'

Assistant thinks: The search results mention several URLs. I want full details from one of them.
Assistant: '<|begin_url|>http://...<|end_url|>'
(System returns full page content)
Assistant: '<|begin_full_page|> ...full page content... <|end_full_page|>'
Now the assistant has enough information and can continue reasoning.
Remember:
- Use '<|begin_search_query|>' to request a web search and end with '<|end_search_query|>'.
- Use '<|begin_url|>' to request full page content and end with '<|end_url|>'.
- When done retrieving information, continue your reasoning.

A.4 Task-Specific Instructions

A.4.1 Open-Domain QA Tasks Instruction

Instruction for Open-Domain QA Tasks

Please answer the following question.
You should provide your final answer in the format \boxed{YOUR_ANSWER}.
Question:
{question}

A.4.2 Math Tasks Instruction

Instruction for Math Tasks

Please answer the following math question.
You should provide your final answer in the format \boxed{YOUR_ANSWER}.
Question:
{question}

A.4.3 Multi-choice Tasks Instruction

Instruction for Multi-choice Tasks

You are to answer the following multiple-choice question by selecting the correct option.
Your final choice should be one of the letters A, B, C, or D. Do not include any answer content beyond the choice letter.
You should provide your final choice in the format \boxed{YOUR_CHOICE}.
Question: {question}

A.4.4 Code Tasks Instruction

Instruction for Code Tasks

Generate a correct Python program that passes all tests for the given problem. You should provide your final code within a Python code block using triple backticks.

```
```python
# YOUR CODE HERE
```
```

Problem Title: {question_title}
Problem Statement:
{question}

A.5 Additional Notes

For all the instructions above, we input them as user prompts, not system prompts. The task-specific instructions in A.4 are used for the QwQ-32B-Preview model. For non-reasoning models like Qwen2.5-32B-Instruct, Qwen2.5-72B-Instruct, and Llama3.3-70B-Instruct, etc., we add a Chain-of-Thought prompt "You should think step by step to solve it." before the question to explicitly make these models reason before giving the final answer.

B Search-o1 Inference Process

B.1 Inference Logic for a Single Question.

For each question, the Search-o1 inference begins by initializing the reasoning sequence with the task instruction I concatenated with the specific question q. As the reasoning model $\mathcal{M}$ generates the reasoning chain $\mathcal{R}$, it may produce search queries encapsulated within the special symbols <|begin_search_query|> and <|end_search_query|>. Upon detecting the <|end_search_query|> symbol, the corresponding search query q_{search} is extracted, triggering the retrieval function `Search` to obtain relevant external documents $\mathcal{D}$. These retrieved documents, along with the reason-in-documents instruction I_{docs} and the current reasoning sequence $\mathcal{R}$, are then processed by the Reason-in-Documents module. This module refines the raw documents into concise, pertinent information r_{final}, which is seamlessly integrated back into the reasoning chain $\mathcal{R}$ within symbols <|begin_search_result|> and <|end_search_result|>. This iterative process ensures that the reasoning model incorporates necessary external knowledge while maintaining coherence and logical consistency, leading to the generation of a comprehensive reasoning chain $\mathcal{R}$ and the final answer a.

B.2 Batch Inference Mechanism.

To efficiently handle multiple questions simultaneously, the Search-o1 framework employs a batch inference mechanism that optimizes both token generation and knowledge refinement. Initially, a set of unfinished reasoning sequences $\mathcal{S}$ is created by concatenating the task instruction I with each question q in the batch $\mathcal{Q}$. The reasoning model $\mathcal{M}$ then generates tokens for all sequences in $\mathcal{S}$ in parallel, advancing each reasoning chain until it either completes or requires external knowledge retrieval. When a search query is identified within any sequence, the corresponding queries

Algorithm 1 Search-o1 Inference

Require: Reasoning Model $\mathcal{M}$, Search function `Search`
Input: Questions $\mathcal{Q}$, Task instruction I, Reason-in-documents instruction I_{docs}
Initialize set of unfinished sequences $\mathcal{S} \leftarrow \{I \oplus q \mid q \in \mathcal{Q}\}$
Initialize set of finished sequences $\mathcal{F} \leftarrow \{\}$
while $\mathcal{S} \neq \emptyset$ **do**
    Generate all sequences in $\mathcal{S}$ until EOS or <|end_search_query|>: $\mathcal{T} \leftarrow \mathcal{M}(\mathcal{S})$ ▷ Batch Generate
    Initialize empty set $\mathcal{S}_r \leftarrow \{\}$ ▷ Reason-in-documents Inputs
    for each sequence Seq $\in \mathcal{T}$ **do**
        if Seq ends with <|end_search_query|> **then**
            Extract search query: $q_{\text{search}} \leftarrow \text{Extract}(\text{Seq}, \text{<|begin\_search\_query|>}, \text{<|end\_search\_query|>})$
            Retrieve documents: $\mathcal{D} \leftarrow \texttt{Search}(q_{\text{search}})$ ▷ Retrieval
            Construct input for Reason-in-documents: $I_{\mathcal{D}} \leftarrow I_{\text{docs}} \oplus \text{Seq} \oplus q_{\text{search}} \oplus \mathcal{D}$
            Append the tuple $(I_{\mathcal{D}}, \text{Seq})$ to $\mathcal{S}_r$
        else if Seq ends with EOS **then**
            Remove Seq from $\mathcal{S}$, add Seq to $\mathcal{F}$ ▷ Sequence Finished
        end if
    end for
    if $\mathcal{S}_r \neq \emptyset$ **then**
        Prepare batch inputs: $\mathcal{I}_r \leftarrow \{I_{\mathcal{D}} \mid (I_{\mathcal{D}}, \text{Seq}) \in \mathcal{S}_r\}$
        Reason-in-documents: $\mathcal{T}_r \leftarrow \mathcal{M}(\mathcal{I}_r)$ ▷ Batch Generate
        for $i \leftarrow \{1, ..., |\mathcal{T}_r|\}$ **do**
            Let $r \leftarrow \mathcal{T}_r[i]$, Seq $\leftarrow \mathcal{S}_r[i]$.Seq
            Extract knowledge-injected reasoning step: $r_{\text{final}} \leftarrow \text{Extract}(r)$
            Update sequence in $\mathcal{S}$: Seq $\leftarrow$ Insert (<|begin_search_result|>, r_{final}, <|end_search_result|>)
        end for
    end if
end while
Output: Finished Sequences $\mathcal{F}$

are extracted and processed in batches through the `Search` function to retrieve relevant documents $\mathcal{D}$. These documents are then collectively refined by the Reason-in-Documents module, which generates the refined knowledge r_{final} for each sequence. The refined knowledge is subsequently inserted back into the respective reasoning chains. Completed sequences are moved to the finished set $\mathcal{F}$, while ongoing sequences remain in $\mathcal{S}$ for further processing. By leveraging parallel processing for both generation and refinement steps, the batch inference mechanism enhances system throughput associated with handling multiple inputs concurrently.

C Tasks and Datasets

The evaluations used in this experiment include the following two categories:

Challenging reasoning tasks: (1) **GPQA** (Rein et al., 2023) is a PhD-level science multiple-choice QA dataset. The questions are authored by domain experts in physics, chemistry, and biology. In our main experiments, we use the highest quality diamond set containing 198 questions, and in Table 3, we use a more comprehensive extended set containing 546 questions to compare with the performance of human experts. (2) **Math benchmarks** include **MATH500** (Lightman et al., 2024), **AMC2023** [1], and **AIME2024** [2]. MATH500 consists of 500 questions from the MATH test set (Hendrycks et al., 2021). AMC2023 and AIME2024 are middle school math competitions covering arithmetic, algebra, geometry, etc., containing 40 and 30 questions respectively. Among these three datasets, MATH500 and AMC are relatively simple, while AIME is more difficult. (3) **LiveCodeBench** (Jain et al., 2024) is a benchmark for evaluating LLMs' coding capabilities, consisting of easy, medium, and hard difficulty problems. It collects recently published programming problems from competitive platforms to avoid data contamination. We utilize problems from August to November 2024, comprising 112 problems.

Open-domain QA tasks: (1) **Single-hop QA datasets: Natural Questions (NQ)** (Kwiatkowski et al., 2019) contains questions from real Google search queries with answers from Wikipedia articles. **TriviaQA** (Joshi et al., 2017) is a large-

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

scale dataset with questions from trivia websites and competitions, featuring complex entity relationships. (2) **Multi-hop QA datasets: HotpotQA** (Yang et al., 2018) is the first large-scale dataset requiring reasoning across multiple Wikipedia paragraphs. **2WikiMultihopQA (2WIKI)** (Ho et al., 2020) provides explicit reasoning paths for multi-hop questions. **MuSiQue** (Trivedi et al., 2022) features 2-4 hop questions built from five existing single-hop datasets. **Bamboogle** (Press et al., 2023) collects complex questions that Google answers incorrectly to evaluate models' compositional reasoning across various domains.

D Baselines

We evaluate our approach against the following baseline methods:

Direct Reasoning: These methods utilize the model's internal knowledge without retrieval. The open-source models include Qwen2.5-32B-Instruct (Qwen et al., 2024), Qwen2.5-Coder-32B-Instruct (Hui et al., 2024), QwQ-32B-Preview (Team, 2024), Qwen2.5-72B-Instruct (Qwen et al., 2024), and Llama3.3-70B-Instruct (Dubey et al., 2024). Closed-source non-proprietary models include DeepSeek-R1-Lite-Preview (**?**), OpenAI GPT-4o (Hurst et al., 2024), and o1-preview (Jaech et al., 2024). Results for open-source models are based on our implementations, while closed-source model results are sourced from their official releases.

Retrieval-augmented Reasoning: These methods retrieve external information to enhance the reasoning process. We consider two retrieval augmentation approaches: **(1) Standard RAG:** Retrieves the top-10 documents for the original question and inputs them alongside the question into the model for reasoning and answer generation. **(2) RAG Agent (RAgent):** Allows the model to decide when to generate queries for retrieval, as detailed in Section 3.3. To manage the length of retrieved documents, inspired by ReAct (Yao et al., 2023), we first retrieve the top-10 snippets during reasoning. The model then decides which URLs to obtain for the full documents when necessary.

E Implementation Details

For the backbone large reasoning model in Search-o1, we utilize the open-sourced QwQ-32B-Preview (Team, 2024). For generation settings, we use a maximum of 32,768 tokens, temperature of 0.7, top_p of 0.8, top_k of 20, and a repetition penalty of 1.05 across all models. For retrieval, we employ the Bing Web Search API, setting the region to US-EN and the top-k retrieved documents to 10. We use Jina Reader API to fetch the content of web pages for given URLs. For all retrieval-based methods, following (Rein et al., 2023), we apply a back-off strategy where, when a final answer is not provided, we use the result from direct reasoning. For baseline models not specifically trained for o1-like reasoning, we apply Chain-of-Thought (CoT) (Wei et al., 2022) prompting to perform reasoning before generating answers. Detailed instructions for all models are provided in Appendix A. All experiments are conducted on eight NVIDIA A800-80GB GPUs.

F Case Study

Tables 5, 6, and 7 present examples of outputs from our Search-o1 model on the GPQA, AMC2023, and HotpotQA datasets, respectively. The model-generated search queries are enclosed within <|begin_search_query|> and <|end_search_query|>, while the refined search results are enclosed within <|begin_search_result|> and <|end_search_result|>. We observe that our Reason-in-Documents mechanism provides coherent information that effectively meets the information needs of the model's current reasoning step and seamlessly integrates with the preceding reasoning process.

Table 5: An example from Search-o1 on GPQA dataset, with special symbols used in the search queries and search results highlighted in green box and blue box, respectively.

Example #1 from GPQA Dataset

Question:
trans-cinnamaldehyde was treated with methylmagnesium bromide, forming product 1.
1 was treated with pyridinium chlorochromate, forming product 2.
3 was treated with (dimethyl(oxo)-l6-sulfaneylidene)methane in DMSO at elevated temperature, forming product 3.
how many carbon atoms are there in product 3? Choices:
(A) 10
(B) 11
(C) 14
(D) 12
Labeled Answer:
B

Model Output by Search-o1

Alright, so I have this problem here about some chemical reactions, and I need to figure out how many carbon atoms are in the final product, which is product 3. Let's break this down step by step.
First, there's trans-cinnamaldehyde, and it's being treated with methylmagnesium bromide to form product 1. I know that methylmagnesium bromide is a Grignard reagent, which is nucleophilic and usually adds to carbonyl compounds. So, trans-cinnamaldehyde has a carbonyl group, right? It's an aldehyde, so it has a C=O bond. The Grignard reagent should attack the carbonyl carbon, adding the methyl group and turning the aldehyde into an alcohol, I think. But let me make sure.
Wait, maybe I should look up what exactly happens when a Grignard reagent reacts with an aldehyde. Let me try to search for that.
<|begin_search_query|>reaction of grignard reagent with aldehyde<|end_search_query|>
<|begin_search_result|>Reacting a Grignard reagent with an aldehyde results in the formation of a secondary alcohol, incorporating one additional carbon atom from the Grignard reagent. Therefore, if trans-cinnamaldehyde (9 carbon atoms) reacts with methylmagnesium bromide (1 carbon atom), the resulting product (product 1) will have 10 carbon atoms. Subsequent protonation to form product 2 and further reactions to form product 3 do not change the carbon atom count, assuming no carbon is added or removed in those steps. Hence, product 3 should have 10 carbon atoms.<|end_search_result|>
Hmm, according to this, product 1 should have 10 carbon atoms, starting from trans-cinnamaldehyde, which has 9 carbon atoms, plus the methyl group from the Grignard reagent, adding one more carbon atom. That makes sense.
Next, product 1 is treated with pyridinium chlorochromate to form product 2. Pyridinium chlorochromate (PCC) is used for oxidations, particularly for oxidizing primary alcohols to aldehydes or secondary alcohols to ketones. Since product 1 is a secondary alcohol (from the previous step), PCC should oxidize it to a ketone. Again, this shouldn't change the number of carbon atoms; it's just changing the functional group.
...
Alternatively, perhaps the initial trans-cinnamaldehyde has more or fewer carbon atoms than I thought. Let me confirm its structure.
<|begin_search_query|>structure of trans-cinnamaldehyde<|end_search_query|>
<|begin_search_result|>Trans-cinnamaldehyde, also known as (E)-cinnamaldehyde, has the molecular formula C9H8O. It consists of a phenyl ring connected to a propenal group via a double bond, forming a conjugated system that makes the molecule planar. The structure comprises 9 carbon atoms, 8 hydrogen atoms, and 1 oxygen atom, with the functional groups indicating its reactivity in chemical transformations.<|end_search_result|>
Yes, confirmed: trans-cinnamaldehyde has 9 carbon atoms.
Now, reacting with methylmagnesium bromide, which provides a methyl group (-CH3), adds one carbon atom, making product 1 have 10 carbon atoms.
Then, oxidation with PCC to convert the alcohol to a ketone doesn't change the carbon skeleton, so product 2 still has 10 carbon atoms.
...
Therefore, starting from 9 carbon atoms in trans-cinnamaldehyde, adding one carbon atom from the Grignard reagent to make 10, and then adding another carbon atom in the cyclopropanation to make 11.
So, the final answer is $\boxed{B}$

Table 6: An example from Search-o1 on AMC2023 dataset, with special symbols used in the search queries and search results highlighted in <|begin_search_query|> and <|begin_search_result|>, respectively.

Example #2 from AMC2023 Dataset

Question:
There is a unique sequence of integers $a_1, a_2, \cdots a_{2023}$ such that

$$\tan 2023x = \frac{a_1 \tan x + a_3 \tan^3 x + a_5 \tan^5 x + \cdots + a_{2023} \tan^{2023} x}{1 + a_2 \tan^2 x + a_4 \tan^4 x \cdots + a_{2022} \tan^{2022} x}$$

whenever $\tan 2023x$ is defined. What is a_{2023}?
Labeled Answer:
-1

Model Output by Search-o1

Alright, so I've got this problem here involving trigonometric identities, specifically with the tangent function. The equation given is:
...
And I need to find the value of a_{2023}.
First off, I know that there are multiple-angle identities for tangent, but I'm not sure if there's a direct formula for such a high multiple like 2023. Maybe there's a general formula for $\tan nx$ in terms of $\tan x$, and perhaps this problem is asking for the coefficients in that expansion.
...
Wait, I remember that tangent of multiple angles can be expressed using the Bernoulli polynomials or something similar, but I'm not sure. Maybe I need to search for a general formula for $\tan nx$.
Let me try to search for "tangent multiple angle formula" to see if there's a general expression.
<|begin_search_query|>tangent multiple angle formula<|end_search_query|>
<|begin_search_result|>The recursive formula for tan(n θ) is given by:

$$\tan(n\theta) = \frac{\tan[(n-1)\theta] + \tan\theta}{1 - \tan[(n-1)\theta]\tan\theta}$$

This formula can be used iteratively to compute tan(n θ) for any integer n, making it particularly useful for large values of n, such as n=2023, as encountered in the problem. This approach avoids the complexity of deriving high-degree polynomial expressions directly and leverages the tangent addition formula in a step-by-step manner.<|end_search_result|>
Okay, so there's a recursive formula for $\tan(n\theta)$:

$$\tan(n\theta) = \frac{\tan[(n-1)\theta] + \tan\theta}{1 - \tan[(n-1)\theta]\tan\theta}$$

That seems useful, especially for large n, like 2023. But I'm not sure how to apply this recursively up to 2023 without going through each step, which isn't practical.
Maybe there's a general expression for $\tan(nx)$ in terms of $\tan(x)$ as a rational function.
...
Looking at $n = 1$:

$$\tan x = \frac{a_1 \tan x}{1} \implies a_1 = 1$$

For $n = 3$, $a_3 = -1$
For $n = 5$, $a_5 = 1$
So it seems that $a_n = (-1)^{(n-1)/2}$ for odd n.
Wait, let's check:
For $n = 1$, $(-1)^{(1-1)/2} = (-1)^0 = 1$, which matches $a_1 = 1$
For $n = 3$, $(-1)^{(3-1)/2} = (-1)^1 = -1$, which matches $a_3 = -1$
For $n = 5$, $(-1)^{(5-1)/2} = (-1)^2 = 1$, which matches $a_5 = 1$
So, generalizing, for odd n, $a_n = (-1)^{(n-1)/2}$
Now, 2023 is odd, so $a_{2023} = (-1)^{(2023-1)/2} = (-1)^{1011} = -1$
Therefore, $a_{2023} = -1$
I think that's the answer.
*Final Answer**

$$\boxed{-1}$$

Table 7: An example from Search-o1 on HotpotQA dataset, with special symbols used in the search queries and search results highlighted in green box and blue box, respectively.

Example #3 from HotpotQA Dataset

Question:
Were Goo Goo Dolls and Echosmith formed in the same city?
Labeled Answer:
no

Model Output by Search-o1

Alright, I've got this question: Ẅere Goo Goo Dolls and Echosmith formed in the same city?Ḧmm, I don't know much about these bands, so I'll need to look that up.
First, I should find out where each band was formed. Maybe start with Goo Goo Dolls. I think they're an older band, but I'm not sure. Let me search for their formation place.
<|begin_search_query|>where was Goo Goo Dolls formed?<|end_search_query|>
<|begin_search_result|>The Goo Goo Dolls were formed in Buffalo, New York, in 1986. They initially performed under the name Sex Maggots but changed to Goo Goo Dolls in 1986 for a gig, choosing the name from an ad in True Detective magazine.
<|end_search_result|>
Okay, so Goo Goo Dolls were formed in Buffalo, New York. Now, I need to find out where Echosmith was formed.
<|begin_search_query|>where was Echosmith formed?<|end_search_query|>
<|begin_search_result|>Echosmith was formed in Chino, California, in February 2009.<|end_search_result|>
Chino, California, huh? That sounds different from Buffalo, New York. But maybe I should double-check to make sure I have the right information.
Let me look up Echosmith again to confirm their formation city.
<|begin_search_query|>Echosmith formation location<|end_search_query|>
<|begin_search_result|>Echosmith was formed in Toluca Lake, Los Angeles, California, in February 2009. The initial confusion about the formation location being Chino, California, or Buffalo, New York, is resolved by multiple recent sources confirming their formation in Los Angeles. Chino might be their hometown, but the band was established in Toluca Lake, Los Angeles.<|end_search_result|>
Wait, now it says Toluca Lake, Los Angeles, California. Earlier, I had Chino, California. Maybe there's some confusion here. I should check more sources to make sure.
<|begin_search_query|>Echosmith origin city<|end_search_query|>
<|begin_search_result|>Echosmith was born in Chino, California, but formed the band in Toluca Lake, Los Angeles, California, in February 2009. Chino is their hometown, while Toluca Lake is where they officially formed as a band.<|end_search_result|>
So, it seems like Echosmith originates from Chino, California, but was formed in Toluca Lake, Los Angeles, California.
Now, comparing that to Goo Goo Dolls, who were formed in Buffalo, New York.
Clearly, Buffalo, New York, is different from Toluca Lake, Los Angeles, California.
But just to be thorough, maybe there's a connection I'm missing.
Let me see if Echosmith has any ties to Buffalo, New York.
<|begin_search_query|>Echosmith Buffalo, New York<|end_search_query|>
<|begin_search_result|>Echosmith has performed in Buffalo, New York, in the past, as evidenced by a setlist from a concert at Waiting Room on November 1, 2013. However, there are no current tour dates listed for Buffalo in the searched web pages.<|end_search_result|>
So, they've played in Buffalo, but that doesn't mean they were formed there.
Let me see if there's any other connection between the two bands.
<|begin_search_query|>Goo Goo Dolls and Echosmith connection<|end_search_query|><|begin_search_result|>The maximum search limit is exceeded. You are not allowed to search.<|end_search_result|>
Hmm, it seems I've reached my search limit. Based on the information I have so far, Goo Goo Dolls were formed in Buffalo, New York, and Echosmith was formed in Toluca Lake, Los Angeles, California. These are different cities, so the answer should be no, they were not formed in the same city.
Therefore, I can confidently say that Goo Goo Dolls and Echosmith were not formed in the same city.
*Final Answer**

$$\boxed{\text{no}}$$