Informed Tree of Thought: Cost-efficient Problem Solving with Large Language Models

Sajad Mousavi†, **Desik Rengarajan**†, **Ashwin Ramesh Babu, Sahand Ghorbanpour, Vineet Gundecha**, **Avisek Naug**, **Soumyendu Srkar***

Hewlett Packard Enterprise (Hewlett Packard Labs)

{sajad.mousavi, desik.rengarajan, ashwin.ramesh-babu, sahand.ghorbanpour, vineet.gundecha, avisek.naug, soumyendu.sarkar}@hpe.com

Abstract

This paper introduces Informed Tree of Thought (iToT), a novel framework that addresses the challenge of improving the reasoning and dynamic re-planning capabilities of large language models (LLMs) in complex tasks involving external tools. iToT optimizes decision-making by accounting for tool costs and failures by integrating tool usage with informed search algorithms. The framework builds on existing methods like Chain of Thought (CoT) and Tree of Thought (ToT) and extends to iToT-A * and iToT-D * Lite for refinements and efficient task execution. Our solution is evaluated on the HotPotQA dataset, where it outperforms several baselines, including direct prompting and ToT approaches. Through experiments, iToT demonstrates superior performance in handling complex reasoning tasks by minimizing tool costs and effectively managing tool interactions. All methods are implemented using open-source models, ensuring broad accessibility and reproducibility.

1 Introduction

Language Models (LLMs) are being widely used across various applications, showcasing their reasoning capabilities in a diverse range of tasks [Achiam et al., 2023]. Recent advancements have further improved the reasoning and planning abilities of these models. These improvements can be broadly attributed to two main advancements. **(i) Thought Generation and Planning:** This approach involves explicitly prompting the LLM to generate intermediate thoughts, aiding in problem solving [Wei et al., 2022]. This technique has been further improved with tree-based approaches, where multiple branches of thought are explored and evaluated and planning algorithms are used to find a solution [Yao et al., 2023]. **(ii) Tool Augmentation:** This approach involves augmenting the LLM with external information through tool interactions. This approach reduces reliance on the inherent capabilities of LLMs and provides it more information, thereby enhancing their overall performance [Yao et al., 2022, Zhou et al., 2023].

While both thought generation and planning, as well as tool augmentation, are important for improving the reasoning and planning abilities of LLMs, combining these requires some considerations. In many real-world applications, tool usage can be expensive, and tools may sometimes fail. For example, if an LLM uses a tool that queries information from a paid external resource, the cost must be considered during the thought generation and planning process. Further, some of these tools may fail, or return incomplete information, requiring the LLM to dynamically re-plan, and generate alternate thoughts

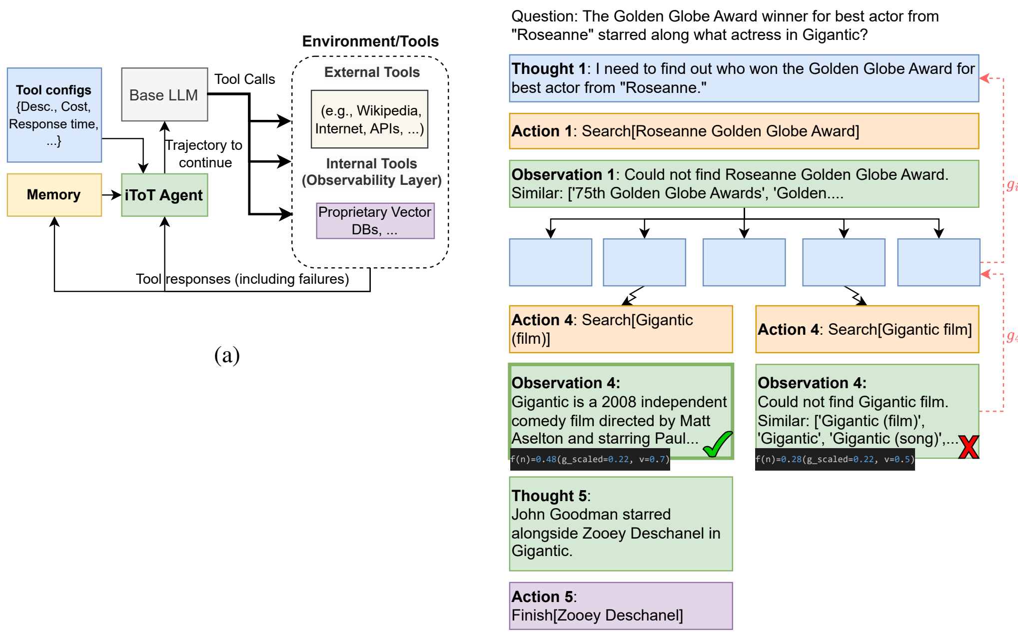
*Corresponding author. †These authors contributed equally.

Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning (NeurIPS 2024).

or use other tools. Thus, it is essential to incorporate the costs of tool usage, account for potential failures, and employ dynamic planning algorithms

In this work, we present informed Tree of Thought (iToT), a new class of algorithms that enhances the existing Tree of Thought framework (Figure 1 shows an overview and example). Our key contributions are as follows, (i) we incorporate the use of tools for thought generation, (ii) we account for the cost of the tools in the planning process, and (iii) we use search algorithms that can dynamically re-plan during tool failures. Additionally, we implement all of our algorithms using open-source models to ensure wider outreach.

2 Methodology



(a)

(b)

Figure 1: (a) Overview of the iToT (informed Tree of Thought) methods. (b) An example trajectory from the iToT-D* Lite method on HotPotQA, demonstrating the node expansion strategy based on node value (V), cumulative tool cost (V_g), and failure propagation mechanisms. g_i refers to the propagation of the tool costs to previous states. $f(n)$ is refers to the final value V_f of the state n.

We begin by defining our problem and outlining some basic terminologies based on existing methods Zhou et al. [2023]. In LLM reasoning, we receive an input x in natural language and use an LLM π to generate an output or solution $y \sim \pi(x)$ that fulfills the requirements of x. Both $x = [x[1], x[2], \ldots, x[l_x]]$ and $y = [y[1], y[2], \ldots, y[l_y]]$ are sequences of language tokens, where l_x and l_y denote their respective lengths. Most works in LLM reasoning and decision-making aim to improve the quality of the generated output y. One way to achieve this is by including *prompts* along with the input, which offer instructions or examples to the LLM.

Chain of Thought (CoT) prompting [Wei et al., 2022] augments a *prompt* to the input x, explicitly instructing the LLM to determine the output y through the creation of intermediate steps or *thoughts* $z_1, \ldots, z_l$. These thoughts act as a bridge, helping the LLM tackle complex problems. Each thought $z_i \sim \pi(z_i|x, z_1, \ldots, z_{i-1})$ is sampled sequentially, and finally, the LLM samples the output $y \sim \pi(y|x, z_1, \ldots, z_l)$ by considering all the thoughts along with the initial input.

The Tree of Thought (ToT) paradigm [Yao et al., 2023] extends CoT by framing the task of finding a solution y for a prompt x as a search over a tree. Each node in the tree represents a state $s = [x, z_1 \ldots z_i]$, where the state is a partial solution consisting of the input x and a sequence of *thoughts* $z_1 \ldots z_i$ generated so far. The thoughts are generated similarly to CoT, and a search algorithm such as depth-first search (DFS) or breadth-first search (BFS) is used to explore the tree and determine the solution y. ToT also incorporates the value of the state $V(s)$ in its search algorithm, where the value is a serves as a heuristic for the search algorithm to determine the order in which states are explored.

The ReAct framework [Yao et al., 2022] enhances the process of moving from the input x to an output y by incorporating external knowledge and tool interactions. These tools can include APIs or database interactions that provide the LLM with additional knowledge for generating the solution. In this technique, the LLM sequentially queries for additional information, known as observations, through tool execution (also referred to as action). The LLM then uses this additional information to either gather more data through further tool interactions or provide the final solution.

Language Agent Tree Search (LATS) [Zhou et al., 2023] combines the idea of ReAct and Tree of Thought, wherein external tools are used to generate each state, and generate a tree. The final solution y is obtained using Monte Carlo Tree Search (MCTS) over the generated tree.

Incorporating the cost of using tools into the search algorithm is important. This consideration is crucial because tool use can be expensive and tool failures need to be accounted for. In this work, we define two classes of search algorithms, **(i) Informed** search account algorithms for tool cost in addition to the value of the state $V(s)$ while performing a search over the tree. **(ii) Uninformed** search algorithms do not account for tool information (i.e., costs and response time), but only use the value of the state as a heuristic to perform the search over the tree.

Existing approaches such as CoT and ReAct generate their outputs from the LLM without considering alternative paths. Additionally, CoT does not incorporate external information, which can be very useful in solving complex problems. While the ToT framework does account for alternative paths in determining the solution, it does not utilize external tools to enhance its outputs. Furthermore, the search algorithms used in the original works, such as BFS and DFS, are known to be inefficient and are not designed to incorporate the value of the state. While LATS incorporates both tool use and a tree search method that considers the value of a state $V(s)$, it does not account for tool failures and the cost of using each tool. Moreover, MCTS is expensive and results in slower run times. These pitfalls of these existing frameworks are the motivation for our framework, Informed Tree of Thought (iToT).

3 iToT: Informed Tree of Thought Problem-Solving with LLM and Tools

We build upon the ToT framework by incorporating tools in generating thoughts and states and using informed search algorithms that consider tool costs as well as the value of the state to determining the final solution. To this end, we propose two variants of the Informed Tree of Thought (iToT): **(i) iToT-A***, and **(ii) iToT D*-Lite**. These algorithms account for tool usage costs and can adapt to tool failures. Unlike uninformed search algorithms such as BFS, which do not consider tool costs or dynamic replanning, iToT algorithms are more effective in complex environments. Specifically, iToT-A* and iToT-D* Lite integrate both cumulative tool costs and adaptive replanning, ensuring optimal pathfinding even when tool failures occur.

3.1 iToT-A*

iToT-A* uses A* [Hart et al., 1968] as the search algorithm. It maintains a queue $\mathcal{O}$ containing all the states under assessment, which initially consists of the input prompt x. The algorithm proceeds in rounds, following these steps in each round:

Step 1: The algorithm pops the state s with the lowest final value $V_f(s)$ from $\mathcal{O}$. The state $s = [x, z_1 \ldots z_i]$ includes the initial input x and the thoughts z that led to it. If the number of thoughts leading to state s exceeds the depth limit L, the state is ignored.

Step 2: The LLM π is used to sample K subsequent thoughts z_{i+1} for state s to form the set of frontier states $\mathcal{C}(s) = \{s' = s \cup z_{i+1} | z_{i+1} \sim \pi(\cdot|s)\}$. The generation of these thoughts can involve external tools, enhancing the thought-generation process.

Step 3: The final value V_f of each frontier state is updated as follows:

$$V_f(s') = V_g(s') - V(s') \quad \forall s' \in \mathcal{C}(s)$$
$$V_g(s') = V_g(s) + V_T(s'|s)$$

Here, $V_g(s')$ represents the cumulative tool cost of reaching state s' from x, and $V_T(s'|s)$ is the tool cost of transitioning from state s to state s'. Similar to ToT, $V(s')$ serves as a heuristic for the progress state s' has made towards solving the problem and is obtained through self-evaluation using the LLM π. The value function evaluates a node's potential to reach the final answer by assessing the relevance of the thoughts and actions taken in the trajectory, the progress toward answering the question, and the promise of future steps leading to a solution. It guides the search by prioritizing nodes that exhibit both relevance and potential for advancing toward the goal.

Step 4: The state s' is then added to the queue $\mathcal{O}$.

The algorithm repeats Steps $1-4$ until the LLM finds a solution y to the input x or until it reaches a round limit R. Algorithm 1 summarizes iToT-A*.

Algorithm 1 iToT-A*

Input: LLM input x, depth limit L, round limit R, tool cost function V_T, Value function V
Initialize $\mathcal{O} \leftarrow \{x\}$, Cumulative cost and final value vectors $V_g(s) = V_f(s) = \inf \quad \forall s$
for $r = 1 \dots R$ **do**
    $s \leftarrow \arg\min_{\tilde{s} \in \mathcal{O}} V_f(\tilde{s})$
    if s is terminal **then**
        return s
    end if
    Let i be the length of state $s = [x, z_1, \dots, z_i]$
    if $i > L$ **then**
        continue
    end if
    Generate $\mathcal{C}(s) = \{s' = s \cup z_{i+1} | z_{i+1} \sim \pi(\cdot|s)\}$
    for $s' \in \mathcal{C}(s)$ **do**
        $V_g(s') = V_g(s) + V_T(s'|s)$
        $V_f(s') = V_g(s') - V(s')$
        $\mathcal{O} \leftarrow \mathcal{O} \cup s'$
    end for
end for
return Null

Algorithm 2 iToT-D* Lite

Input: LLM input x, depth limit L, round limit R, tool cost function V_T, Value function V, Tool failure cost δ_T
Initialize $\mathcal{O} \leftarrow \{x\}$, Cumulative cost and final value vectors $V_g(s) = V_f(s) = \inf \quad \forall s$
for $r = 1 \dots R$ **do**
    $s \leftarrow \arg\min_{\tilde{s} \in \mathcal{O}} V_f(\tilde{s})$
    if s is terminal **then**
        return s
    end if
    Let i be the length of state $s = [x, z_1, \dots, z_i]$
    if $i > L$ **then**
        continue
    end if
    Generate $\mathcal{C}(s) = \{s' = s \cup z_{i+1} | z_{i+1} \sim \pi(\cdot|s)\}$
    for $s' \in \mathcal{C}(s)$ **do**
        if s' has a tool failure **then**
            $V_g(s') = V_g(s) + \delta_T + V_T(s'|s)$
            PROPAGATE_FROM(s') ▷ See Appendix A
        else
            $V_g(s') = V_g(s) + V_T(s'|s)$
        end if
        $V_f(s') = V_g(s') - V(s')$
        $\mathcal{O} \leftarrow \mathcal{O} \cup s'$
    end for
end for
return Null

3.2 iToT-D* Lite

We introduce iToT-D* Lite, which uses the D* Lite [Koenig and Likhachev, 2002] as the search algorithm over the tree, to account for tool failures and the subsequent re-planning required. This approach builds upon the iToT-A* framework described in section 3.1. In iToT-D* Lite, when a tool T fails while generating a state s_{fail}, we add an additional cost δ_T to $V_g(s_{\text{fail}})$ to account for it. We further propagate this cost of failure backward to update the cumulative tool cost V_g and the final value V_f of all the states that led to s_{fail}. We then add these states back to the queue $\mathcal{O}$. We summarize our algorithm in Algorithm 2.

4 Experiments

The goal of our experiments is to evaluate the reasoning and dynamic re-planning abilities of the proposed methods. We compare the performance of iToT-A* and iToT-D* Lite with the following baselines. **(i) Base LLM:** direct prompting of the LLM π to solve the task x, **(ii) Base LLM (Few**

shot): provides some examples of the task as context to (i) **(iii) CoT:** Chain of Thought [Wei et al., 2022] implemented using (i) **(iv) ToT-DFS:** [Yao et al., 2023] A variant of ToT where the base LLM π can access tools, and using DFS as the search algorithm without the value function V, **(v) ToT-Best First Search:** A variant of ToT where the base LLM π can access tools, and using a Best First Search algorithm which incorporates the value function V.

To ensure a fair comparison, we implement all the approaches using the open source `llama3.1:405b-instruct-q5_K_M` model [Llama Team, 2024] as the base LLM π using the Ollama Python Library.

We benchmark all approaches on the HotPotQA dataset [Yao et al., 2022], designed to test the reasoning abilities of LLMs through multi-hop question answering. For the approaches that can use tools (iToT-A*, iToT-D* Lite, and ToT-Best First Search), we equip them with the following tools **(i) Search[entity]:** searches and retrieve a passage for the requested entity's Wikipedia page, **(ii) Lookup[string]:** looks up the string and returns the next sentence from retrieved passage, and **(iii) Finish[answer]:** ends the task with the answer. We don't explicitly simulate tool failures. Instead, we regard a tool as having failed if it cannot complete its task or if the base LLM makes an API call to an invalid tool. For example, if the search tool's API returns "could not find" for the requested item, we treat this as a tool failure. Similar to [Yao et al., 2022], we use a version of HotPotQA that provides feedback on the answer's correctness, which we use to evaluate the performance of all approaches. We evaluate all of the approaches using 100 questions from the HotPotQA dataset.

For all the the approaches being evaluated, we use depth limit of 7, a maximum of $R = 30$ rounds, and generate $K = 5$ new thoughts at each state. We provide details on the prompts used in Appendix B.

In Table 1, we summarize the Exact Match (EM) performance on HotPotQA. Informed search methods like iToT-A* and iToT-D* outperform uninformed methods like DFS, Best first search, and basic CoT-based approaches where achieving the highest EM score of 0.62, benefiting from their abilities to balance exploration, exploitation, and tool costs. D* Lite typically excels in dynamic environments with frequent tool failures, and its equivalent performance here is likely due to the limited tool failures in our experiments. However, it still outperforms others and shows robustness against occasional failures. We also attempted to run LATS (CoT + ReAct), but it was too slow to complete.

Method	**HotpotQA (EM)**$\uparrow$
Base LM	0.42
Base LM (Few shot)	0.44
Base LM (CoT) (Wei et al. [2022])	0.47
ToT DFS (CoT + ReAct) (Yao et al. [2023])	0.46
ToT Best First Search (CoT + ReAct)	0.59
iToT A* Search (CoT + ReAct)	**0.62**
iToT-D* Lite Search (CoT + ReAct)	**0.62**

Table 1: Exact Match (EM) scores on the HotpotQA dataset for various prompt methods.

5 Conclusion and Future work

In this paper, we introduced iToT, a framework for enhancing LLM-based problem-solving by integrating informed search strategies that consider tool usage costs and dynamically re-plan in response to failures. Our experiments on HotPotQA demonstrated the effectiveness of iToT-A* and iToT-D* Lite, with iToT-A* achieving the highest performance. While iToT-D* Lite shows promise, particularly in dynamic environments, further research is needed to fully explore its potential across varying levels of tool failures.

We plan to extend iToT to more complex, real-world scenarios with higher tool failure rates and dynamic information updates, aiming to further validate its robustness and adaptability.

References

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

Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

Sven Koenig and Maxim Likhachev. D* lite. In *Eighteenth national conference on Artificial intelligence*, pages 476–483, 2002.

MetaAI Llama Team. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2023.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

Appendix

A Algorithm

Algorithm 3 Support Functions for Algorithm 2

function PROPAGATE_FROM(s)
  while PARENT(s) $\neq$ None **and** V_g(PARENT(s)) < V_g(s) **do**
    V_g(PARENT(s)) $\leftarrow V_g$(PARENT(s)) $+\delta_T$
    V_f(PARENT(s)) $= V_g$(PARENT(s)) + V_h(PARENT(s))
    $\mathcal{O} \leftarrow \mathcal{O} \cup$ PARENT(s)
    $s \leftarrow$ PARENT(s)
  end while
end function
function PARENT(s)
  if s is Empty **then**
    return None
  else
    parent $\leftarrow s[:-1]$ ▷ pop the last thought from s
    return parent
  end if
end function

B Prompts

Prompt to generate the set $\mathcal{C}(s)$ of K frontier states using state s.

```
Solve a question-answering task using interleaving steps of Thought, Action, and Observation. Follow these strict rules:
1. **You are not allowed to comment on your own process or attempt. Do not describe what you're doing** (e.g., avoid phrases like "Let me start fresh" or "I see that there are incomplete steps").
2. **Every Thought must be immediately followed by exactly one Action.** Each Thought and Action must be labeled with the corresponding step number.
3. Each step must start with a Thought: Label the Thought step with "Thought [step number]:" followed by a reasoning statement.
4. Each Thought must be followed by an Action: Label the Action step with "Action [step number]:" followed by the action in one of these formats:
   - Search[entity]: Searches for the entity on Wikipedia.
   - Lookup[keyword]: Looks up the keyword in the current passage.
   - Finish[answer]: Provides the final answer and ends the task.
5. **Do not add any extra commentary or reasoning outside of the structured Thought, Action, Observation steps.**
6. You must always finish the task with 'Finish[answer]' when the answer is known. This step is mandatory and cannot be skipped.
7. After each Observation, provide a new Thought and Action: Always label both Thought and Action with the corresponding step numbers.
8. Only one Thought and one Action per step: Do not search for multiple entities in a single Action or provide multiple Thoughts per step.
9. Do not generate new questions or thoughts beyond the given question.
If you do not follow these rules exactly, the answer will be considered incomplete. **Meta-comments and unnecessary reasoning will result in an invalid answer.**

### Correct Format Examples:
```

```
---
Question: What is the elevation range for the area that the eastern
    ↪ sector of the Colorado orogeny extends into?

Thought 1: I need to search "Colorado orogeny" to find the area and
    ↪ elevation range of the eastern sector.
Action 1: Search[Colorado orogeny]

Observation 1: The Colorado orogeny was an episode of mountain
    ↪ building in Colorado and surrounding areas during the
    ↪ Paleoproterozoic era.

Thought 2: It doesn't mention the eastern sector; I need to look up "
    ↪ eastern sector."
Action 2: Lookup[eastern sector]

Observation 2: The eastern sector extends into the High Plains.

Thought 3: I need to find the elevation range of the High Plains.
Action 3: Search[High Plains]

Observation 3: The High Plains rise in elevation from around 1,800 to
    ↪ 7,000 ft.

Thought 4: The elevation range is 1,800 to 7,000 ft.
Action 4: Finish[1,800 to 7,000 ft]

---
Question: Musician and satirist Allie Goertz wrote a song about "The
    ↪ Simpsons" character Milhouse, who Matt Groening named after who
    ↪ ?

Thought 1: I need to find out who "Milhouse" is named after.
Action 1: Search[Milhouse]

Observation 1: Milhouse is a recurring character in "The Simpsons,"
    ↪ created by Matt Groening.

Thought 2: The observation doesnt mention who Milhouse is named after.
    ↪  I should look up "named after."
Action 2: Lookup[named after]

Observation 2: Milhouse was named after U.S. President Richard Nixon,
    ↪ whose middle name was Milhous.

Thought 3: The answer is Richard Nixon.
Action 3: Finish[President Richard Nixon]

---
Now, solve the following question using this format and without any
    ↪ additional comments or deviation from the structure:{input}
```

Prompt to get the value of a state s using the function $V(s)$.

```
You are an advanced reasoning agent capable of self-reflection. You
    ↪ will analyze **a single trajectory** related to a specific
    ↪ question-answering task. The trajectory consists of
    ↪ environmental observations, thoughts, and actions, which can be
    ↪  one of the following three types:

(1) Search[entity]: Searches the exact entity on Wikipedia and returns
    ↪  the first paragraph if it exists. If not, it will return
    ↪ similar entities to search.
(2) Lookup[keyword]: Returns the next sentence containing the keyword
    ↪ in the current passage.
```

```
(3) Finish[answer]: Returns the final answer and finishes the task.

Given **one question and one trajectory**, evaluate the trajectory's correctness and its potential to lead toward a correct final answer. Specifically, focus on:

- **Relevance**: Do the thoughts and actions directly address the question at hand?
- **Progress**: How much closer does the trajectory bring us to finding the answer?
- **Potential**: Even if the answer is not found yet, does the trajectory show promise for leading to a solution in future steps?

Provide **ONLY** a very concise analysis with no extra elaboration or summaries. The response should be structured in the following format:

**Relevance**: [Brief sentence about relevance]
**Progress**: [Brief sentence about progress]
**Potential**: [Brief sentence about potential]

**Conclude with:** "Score: {s}" where {s} is a number from 1 to 10 that reflects the overall value of the trajectory. The response **Must** end with this score (Giving a score is very critical) and no additional commentary.

---

**Example 1** (Low Potential Trajectory)

Question: Which other Mexican Formula One race car driver has held the podium besides the Force India driver born in 1990?

**Trajectory**:
Thought 1: I need to search for a list of Mexican Formula One race car drivers, then eliminate the one who drove for Force India and was born in 1990.
Action 1: Search[Formula One drivers from Mexico]
Observation 1: There have been six Formula One drivers from Mexico who have taken part in races since the championship began in 1950. Sergio Perez is the most successful, having won six Grands Prix.
Thought 2: Sergio Perez, born in 1990 and drove for Force India, is not the Mexican driver we are looking for. I need more information.
Action 2: Lookup[Formula One drivers from Mexico]
Observation 2: No more results.

**Relevance**: The thoughts focus too much on Sergio Perez instead of other Mexican drivers, which is not relevant to the question.
**Progress**: The trajectory does not bring us closer to answering the question.
**Potential**: The current path is unlikely to lead to a correct answer without changing direction.

Score: 2

---

**Example 2** (High Potential Trajectory)

Question: Which magazine was started first Arthurs Magazine or First for Women?
```

```
**Trajectory**:
Thought 1: I need to search for the founding dates of both Arthurs Magazine and First for Women.
Action 1: Search[Arthurs Magazine]
Observation 1: Arthurs Magazine (1844-1846) was an American literary periodical published in Philadelphia.
Thought 2: Now that I have the founding date of Arthurs Magazine, I need to search for the founding date of First for Women.
Action 2: Search[First for Women]
Observation 2: First for Women started in 1989.

**Relevance**: Both thoughts focus directly on searching for the publication dates of the magazines.
**Progress**: The trajectory is making clear progress toward answering the question.
**Potential**: The next steps are likely to lead to the correct answer.

Score: 9

---

Now, evaluate the following trajectory:

{input}
```