

LLM Agent Planning

CONTENTS

1

核心介紹

2

任務分解

3

規劃選擇

4

反思改進

5

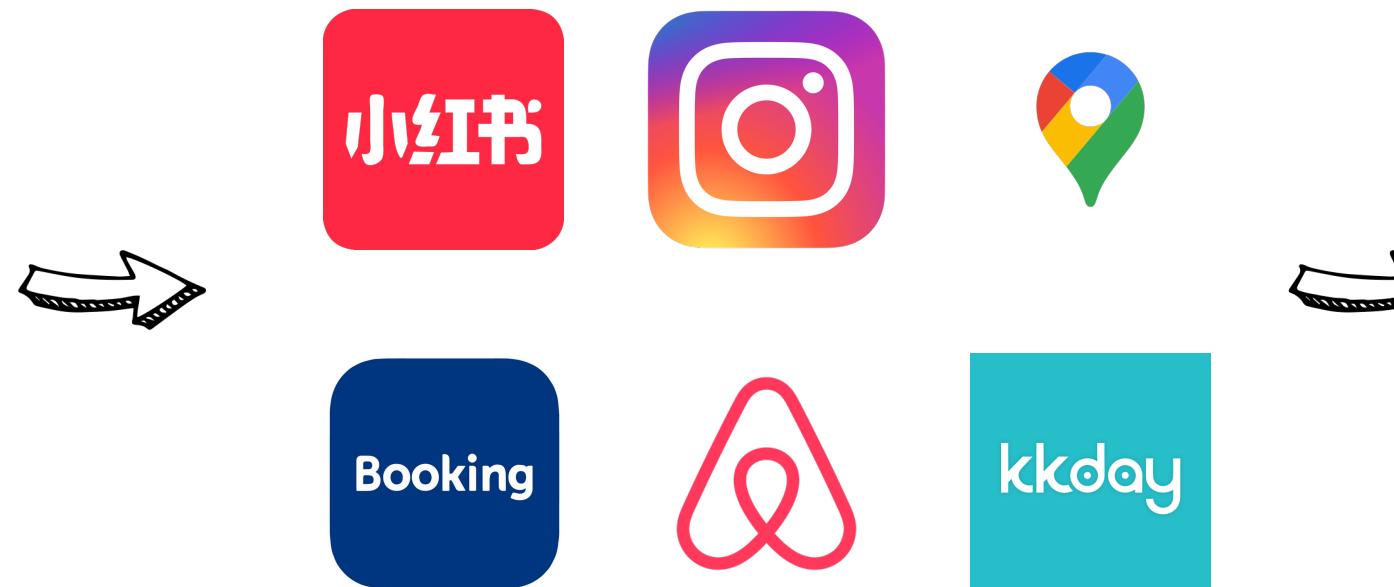
深度思考

01

Introduction

Let's Go to Hawaii

- 1 決定目的地、天數
- 2 安排每日行程
- 3 訂機票
- 4 訂住宿
- 5 確認交通方式
- 6 訂門票
- 7 申請簽證、國際駕照
- ...

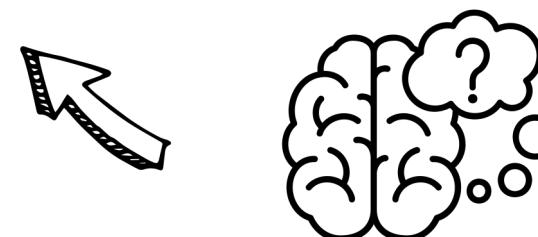


Day 1

- 首爾轉機去歐胡島
- 住海灘附近的飯店
- 買一日票搭公車 ...

Day 5

- 搭飛機去夏威夷島
- 在機場附近租車 ...



六月的機票比七月便宜?

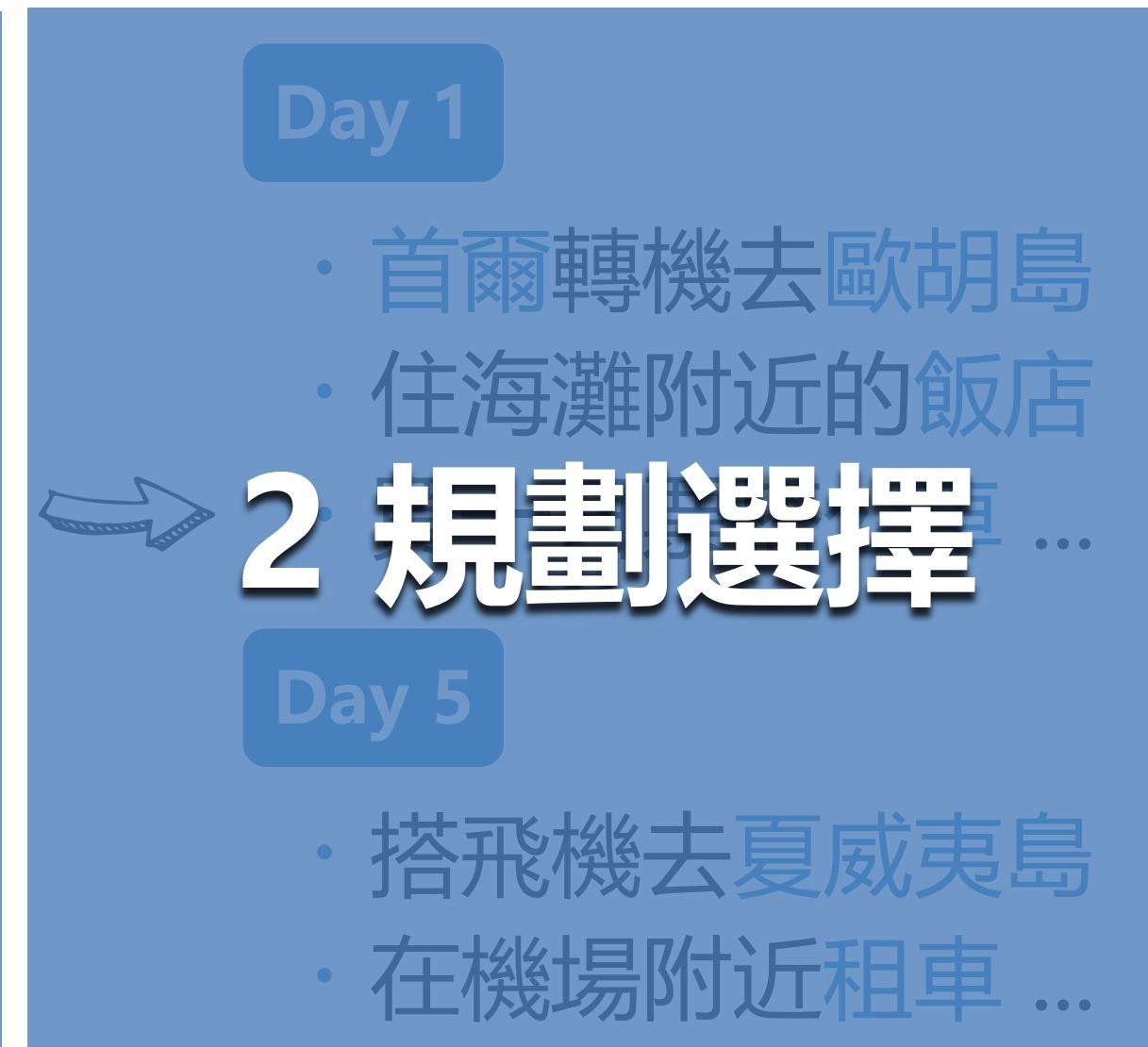
每週五海灘附近會放煙火?

浮潛的景點只有週一週二開放?



偏好：多一點自然風景、水上活動，少一點逛街購物

Planning 的五大核心



Planning 的五大核心

- 1 決定目的地、天數
 - 2 安排每日行程
 - 3 訂機票
 - 4 訂住宿
確認交通工具方式
 - 5 訂門票
 - 6 申請簽證、國際駕照
 - 7 ...
- ## 1 任務分解



- Day 1
- 首爾轉機去歐胡島
 - 住海灘附近的飯店
- 2 規劃選擇
- Day 5
- 搭飛機去夏威夷島
 - 在機場附近租車 ...



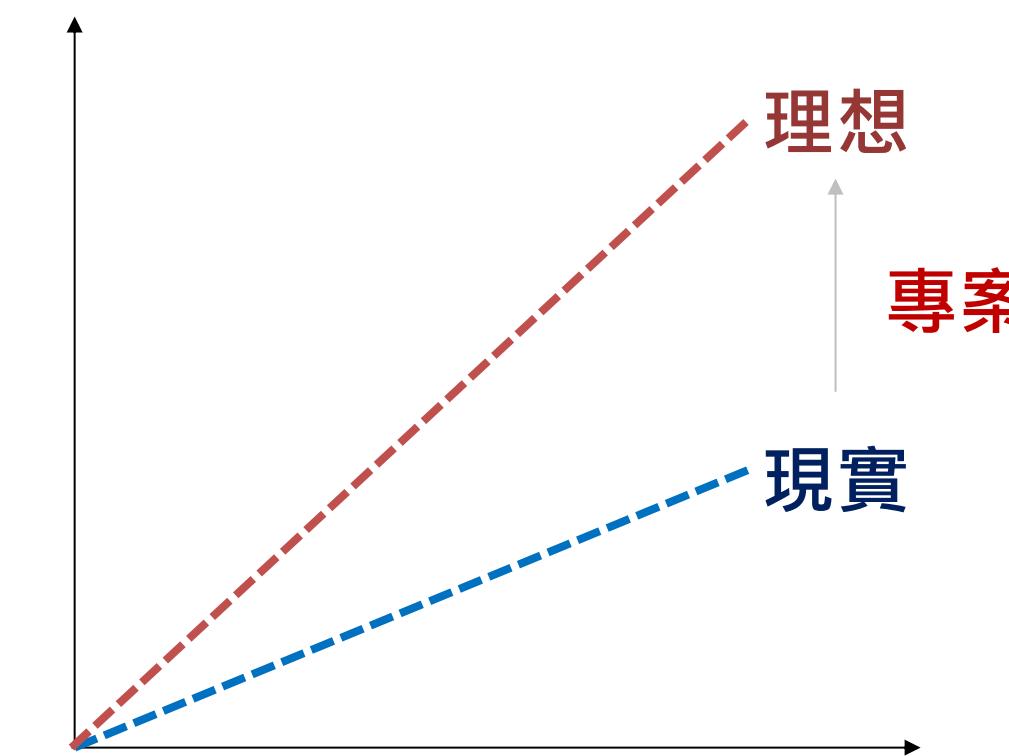
02

Task Decomposition

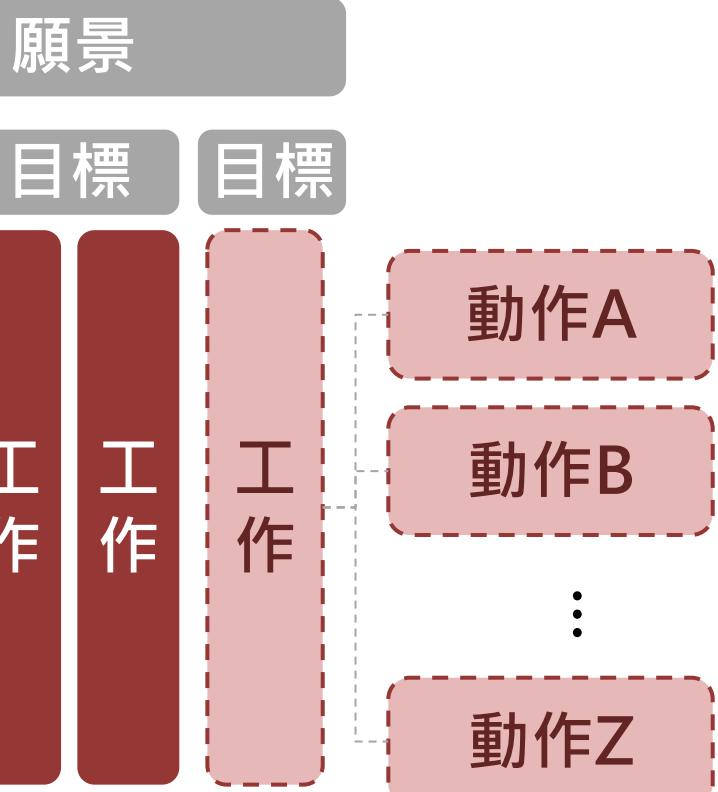
任務分解 = 將複雜任務分解成小任務，各個擊破

現實世界的環境
是動態變化且複雜的

想要透過一次性規劃
來解決是艱鉅的挑戰



日常大家做專案
也是一樣的狀況



先分解，再規劃

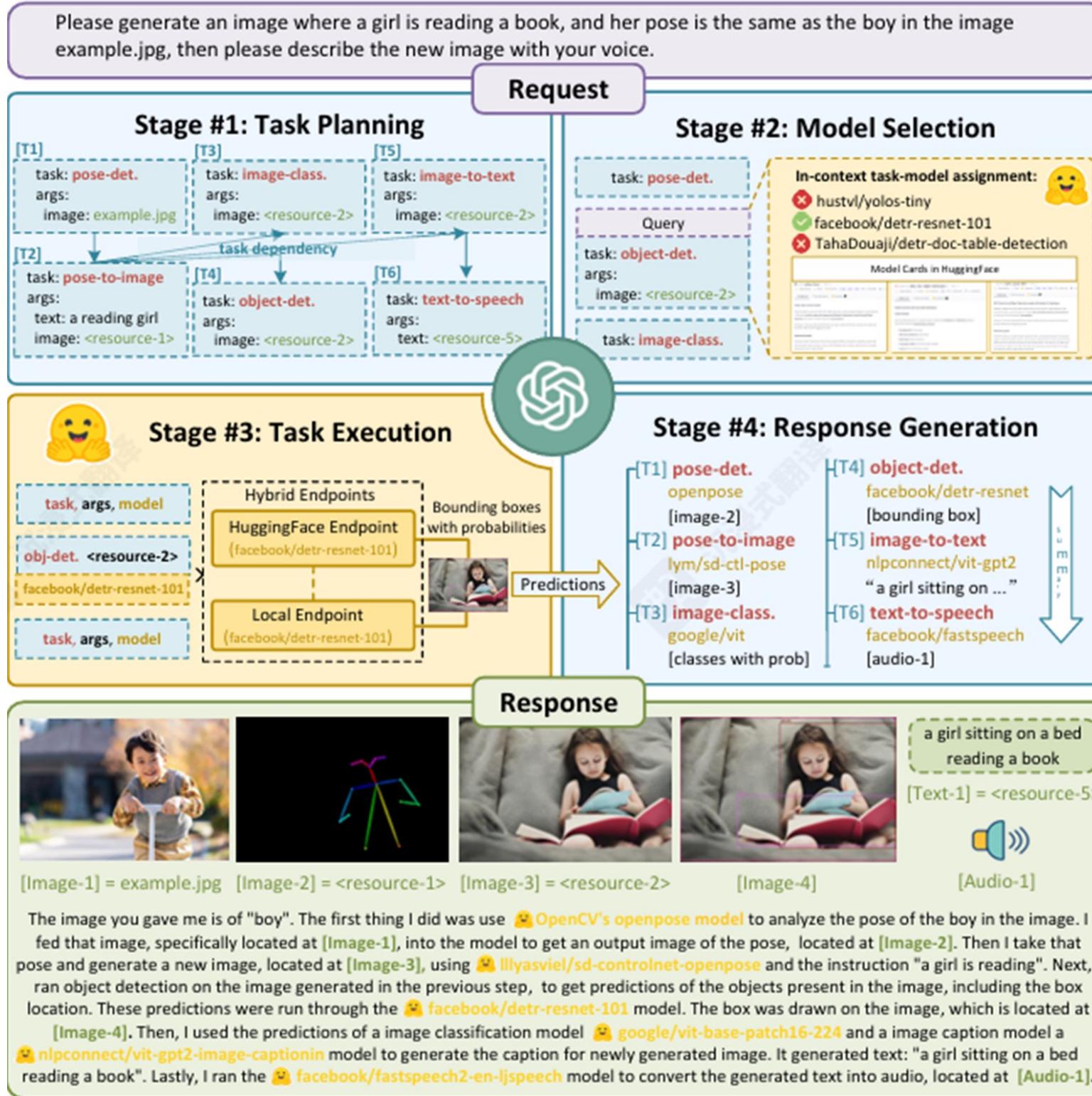
任務分解的類型

| 類型 | 優先分解法 (Decom. First) | 交錯分解法 (Interleaved Decomp.) | 自適應分解法 (Adaptive Decomp.) |
|-----|---|--|---|
| 特色 | #想好再做 #思想巨人 #行動侏儒 #瀑布式 | #邊想邊做 #動態調整 #偏離重點 #敏捷式 | #邊想清楚邊做 #不斷評估 #我全都要 #混合式 |
| 代表作 | <ul style="list-style-type: none">• Input-Output• HuggingfaceGPT | <ul style="list-style-type: none">• CoT 家族• ReAct | <ul style="list-style-type: none">• ReWOO• ReMSV |

任務分解的類型

| 類型 | 優先分解法 (Decom. First) | 交錯分解法 (Interleaved Decomp.) | 自適應分解法 (Adaptive Decomp.) |
|-----|---|--|---|
| 特色 | #想好再做 #思想巨人 #行動侏儒 #瀑布式 | #邊想邊做 #動態調整 #偏離重點 #敏捷式 | #邊想清楚邊做 #不斷評估 #我全都要 #混合式 |
| 代表作 | <ul style="list-style-type: none">• Input-Output• HuggingfaceGPT | <ul style="list-style-type: none">• CoT 家族• ReAct | <ul style="list-style-type: none">• ReWOO• ReMSV |

HuggingfaceGPT



使用 ChatGPT 分析使用者的請求，以理解其意圖並將其拆解為可能可解的任務。

1. 將任務拆解，判斷相依性
2. 挑選適合的模型
3. 調用模型後執行
4. 整合執行結果

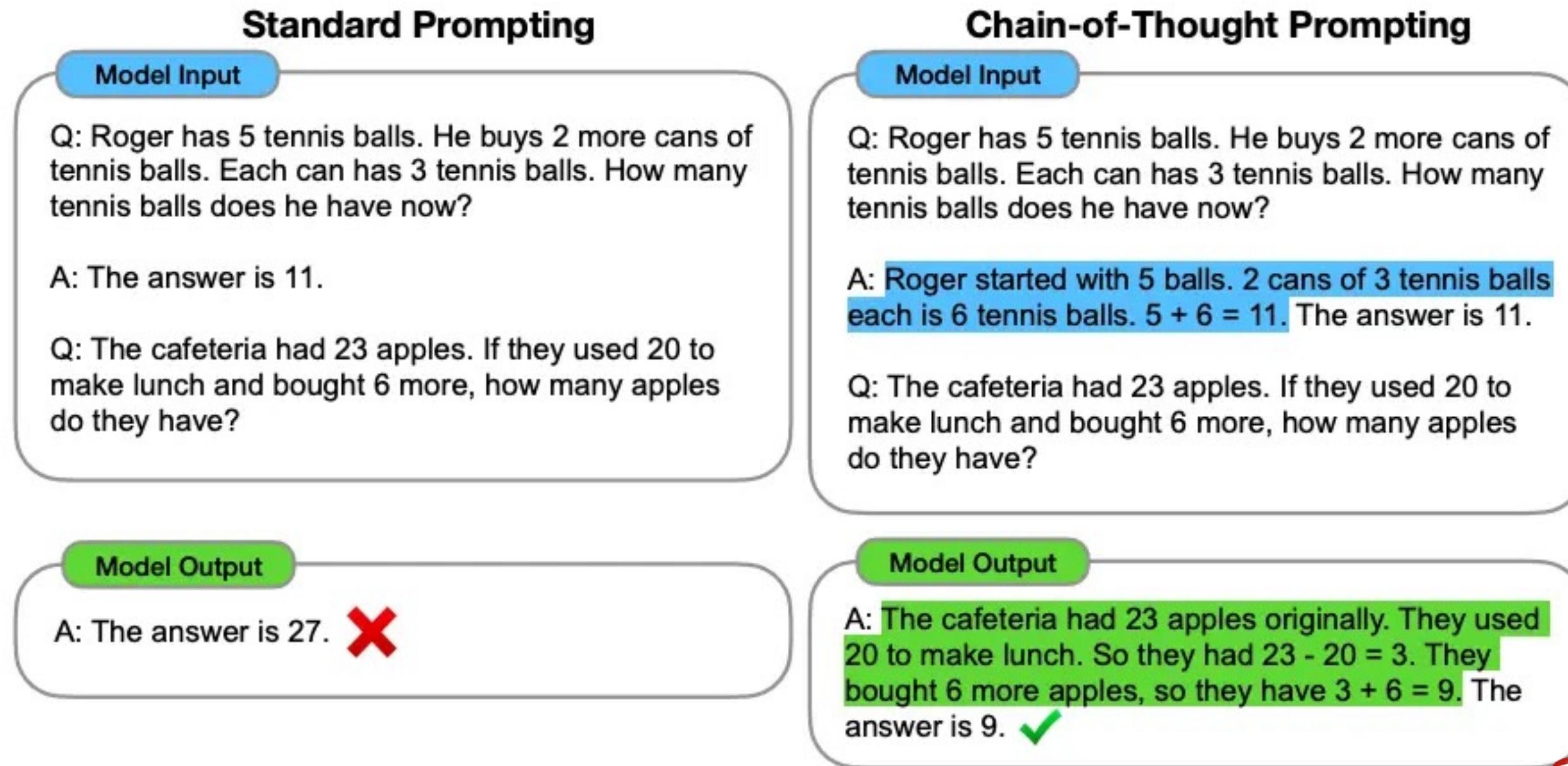


採用全局規劃策略
一個查詢解決全部的事情
若某步驟生成錯誤結果，整個工作流可能會進入無限循環

任務分解的類型

| 類型 | 優先分解法 (Decomp. First) | 交錯分解法 (Interleaved Decomp.) | 自適應分解法 (Adaptive Decomp.) |
|-----|---|--|---|
| 特色 | #想好再做 #思想巨人 #行動侏儒 #瀑布式 | #邊想邊做 #動態調整 #偏離重點 #敏捷式 | #邊想清楚邊做 #不斷評估 #我全都要 #混合式 |
| 代表作 | <ul style="list-style-type: none">• Input-Output• HuggingfaceGPT | <ul style="list-style-type: none">• CoT 家族• ReAct | <ul style="list-style-type: none">• ReWOO• ReMSV |

Chain-of-Thought (CoT) 思維鏈

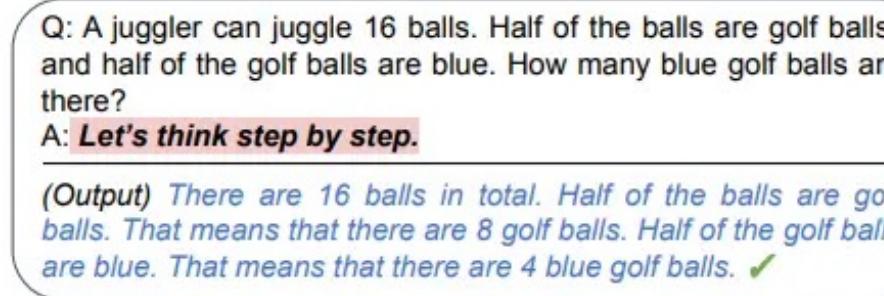


在提示詞中提供範例，讓模型能夠理解思考過程



CoT 家族

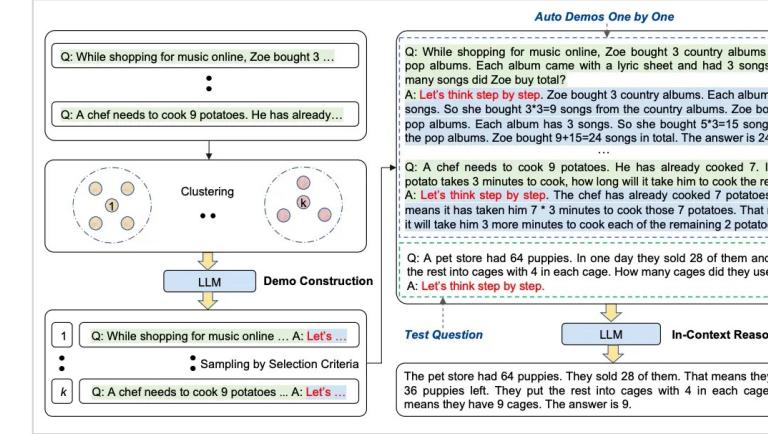
懶得教老師



Zero-shot CoT

每次都要我教才會！啊你是不會自己想喔！
疑？怎麼好像罵一罵就會了？

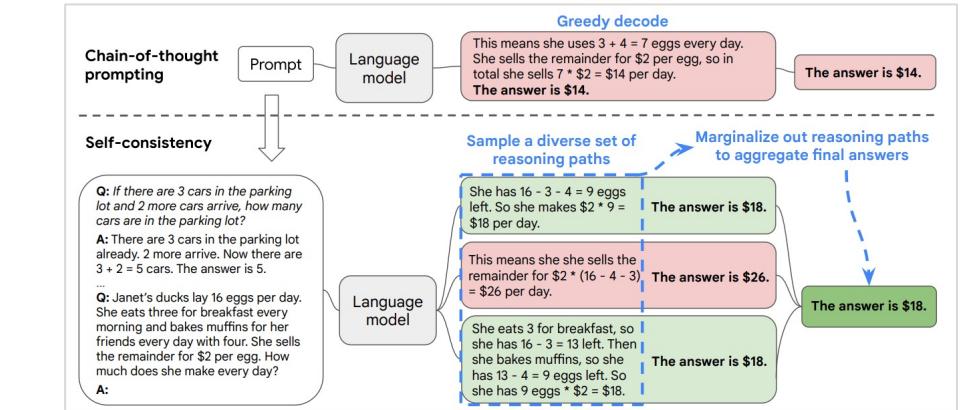
引導式家教



Auto-CoT

先將問題分類 · 再給一個代表性的案例
透過引導式教學，可以有效減輕推論錯誤的狀況

哲學/數學 教授

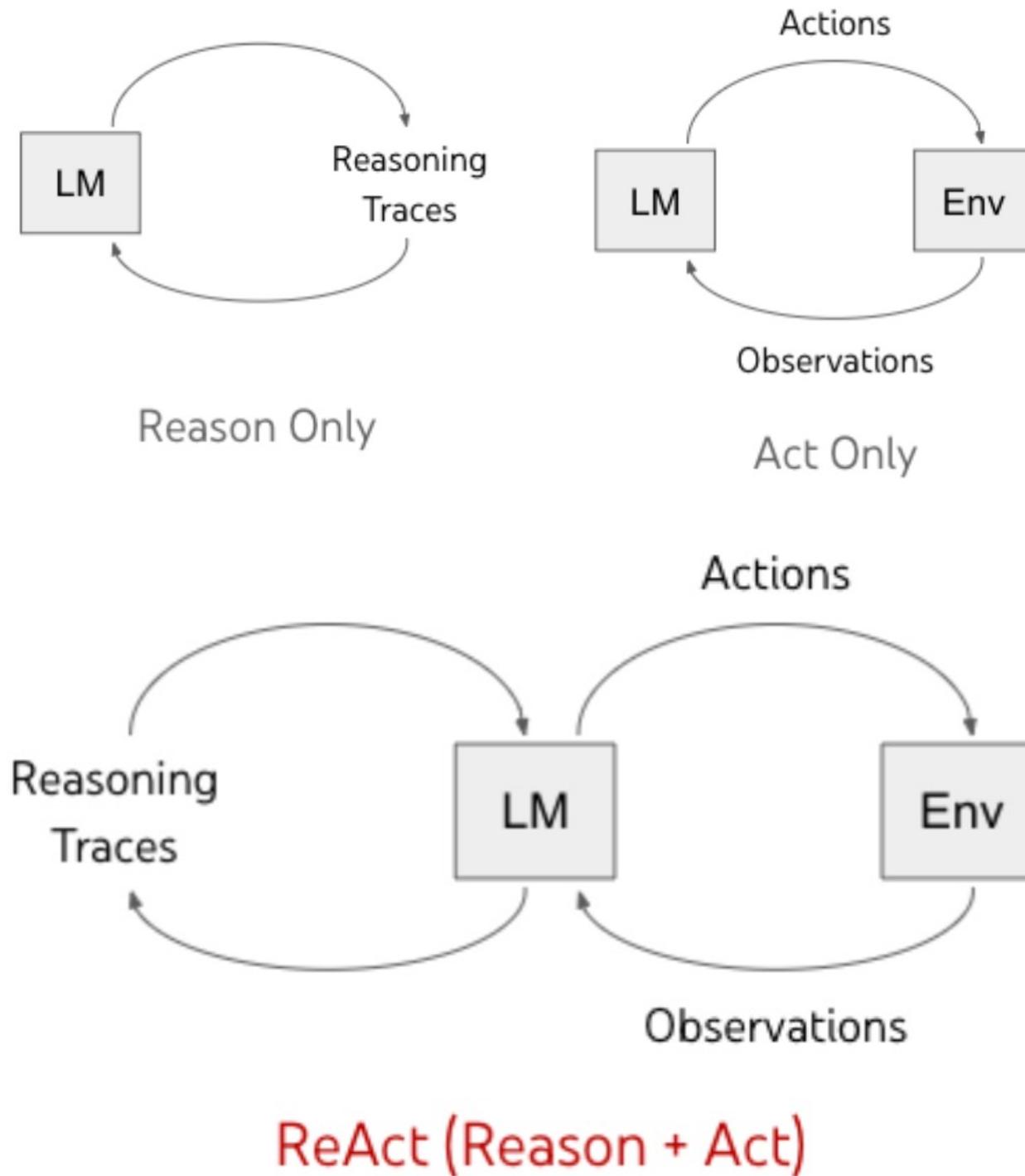


SC-CoT

大學數學老師最愛考的證明題
讓一題多解，找出最有一次答案出現的結果

1. 仰賴模型自己的判斷
2. 模型判斷依據 = 訓練資料 = 與現實可能有差異？
3. 你在那邊想半天，要不要直接做比較快？

ReAct = Reason + Act



思考 + 行動 + 觀察 = 敏捷

1. 讓模型知道如何從行動中獲取訊息
2. 如何搜尋檢索到重要的資訊很重要(Tools)

ReAct = Reason + Act

啟動ReAct模式

你會在「思考 (Thought) → 行動 (Action) → 觀察 (Observation) 」的循環中運作。

在每個循環結束時，你需要輸出一個最終答案 (Answer) 。

- 使用「Thought」來描述你對所被提問問題的想法。
- 使用「Action」來執行你可使用的其中一個動作。
- 「Observation」則是你執行該動作後得到的結果。

你可以使用以下這些動作：

給他工具

call_google :

例如：call_google: Hawaiian bikini models 透過 Google 搜尋 "Hawaiian bikini models" 並返回摘要。

如果你有機會這麼做，或者你對問題不確定，可以查詢 Google 。

Few-shot

範例一：

問題：夏威夷有沒有辣妹？

思考：我可以用 Google 查詢關於夏威夷辣妹的資料

行動：call_google: Hawaiian hot girls 你將會再次被呼叫並收到這個：

觀察：搜尋結果顯示夏威夷當地受歡迎的比基尼模特兒、社群媒體網紅與旅遊相關影像...

你接著輸出：答案：有，夏威夷有很多辣妹，包含比基尼模特兒。

任務分解的類型

| 類型 | 優先分解法 (Decomp. First) | 交錯分解法 (Interleaved Decomp.) | 自適應分解法 (Adaptive Decomp.) |
|-----|---|--|---|
| 特色 | #想好再做 #思想巨人 #行動侏儒 #瀑布式 | #邊想邊做 #動態調整 #偏離重點 #敏捷式 | #邊想清楚邊做 #不斷評估 #我全都要 #混合式 |
| 代表作 | <ul style="list-style-type: none">• Input-Output• HuggingfaceGPT | <ul style="list-style-type: none">• CoT 家族• ReAct | <ul style="list-style-type: none">• ReWOO• ReMSV |

ReWOO (Reasoning WithOut Observation)

透過規劃者規劃藍圖、工作者使用工具、解決者綜合判斷，以生成最終結果

LLM Agent Planning

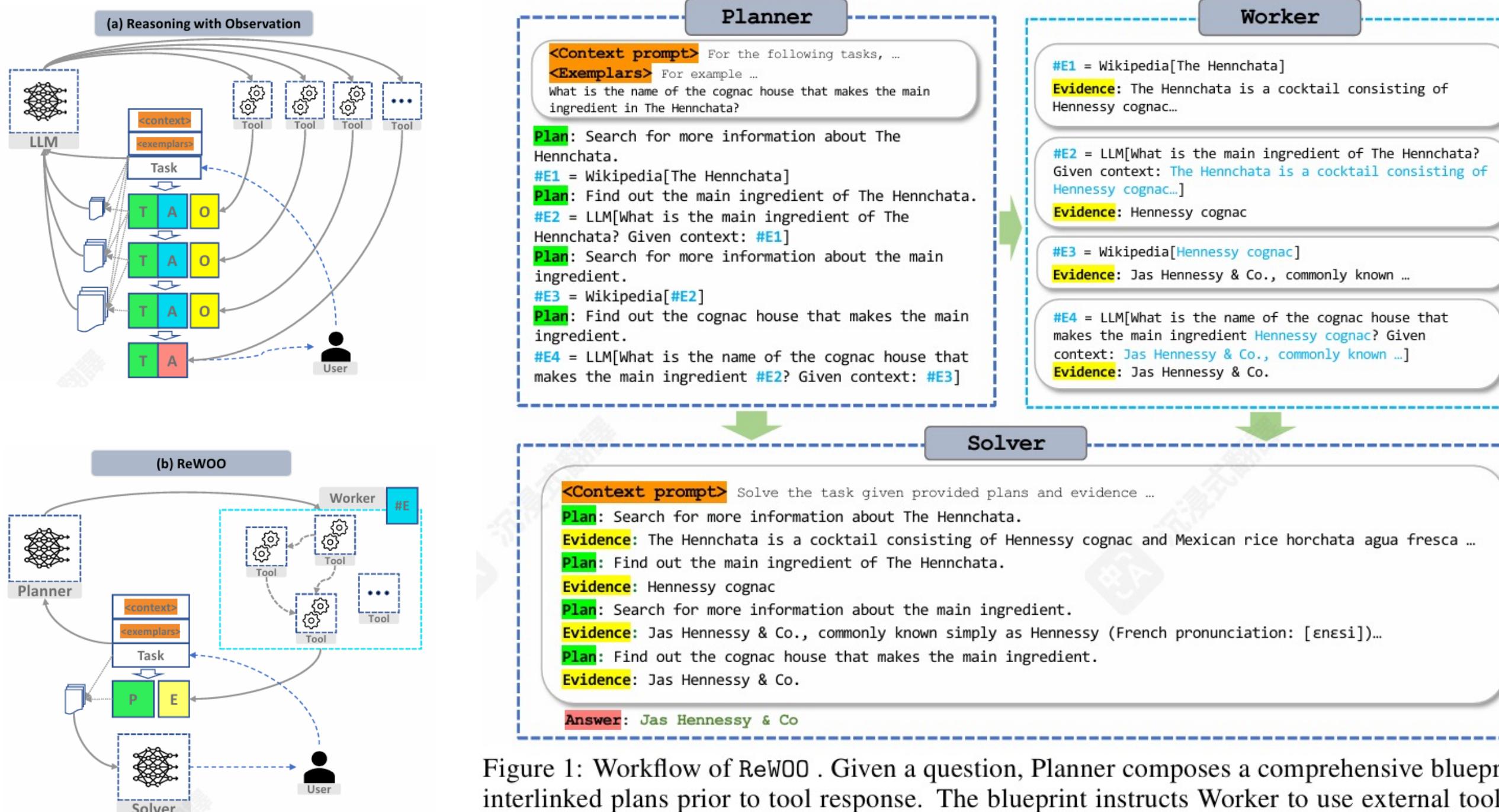


Figure 1: Workflow of ReWOO . Given a question, Planner composes a comprehensive blueprint of interlinked plans prior to tool response. The blueprint instructs Worker to use external tools and collect evidence. Finally, plans and evidence are paired and fed to Solver for the answer.

ReMSV (Reasoning Multiple StATEGY and VOting)

省下來的成本要幹嗎？多做一點事。



怎麼選擇？



| 能力/特性 | 優先分解法 | 交錯分解法 | 自適應分解法 |
|----------|-------|--------|---------|
| 任務順序靜態 | ✓ 是 | ✗ 否 | ✗ 否 |
| 可依工具回饋調整 | ✗ 否 | ✓ 是 | ✓ 是 |
| 可根據記憶調整 | ✗ 否 | ☑ 有限支援 | ✓ 是 |
| 策略可變 | ✗ 否 | ✗ 否 | ✓ 是 |
| 技術實作複雜度 | ★ (低) | ★★ (中) | ★★★ (高) |

03

Plan Selection

如果說...

CoT是走一條單行道路

ToT是走一個十字路口

探索多條路徑

選擇最有希望方向前進

思維樹 (Tree of Thought, ToT) 關鍵組成

① 節點

單個想法

#切入點

③ 探索

腦力激盪

#推演

② 分支

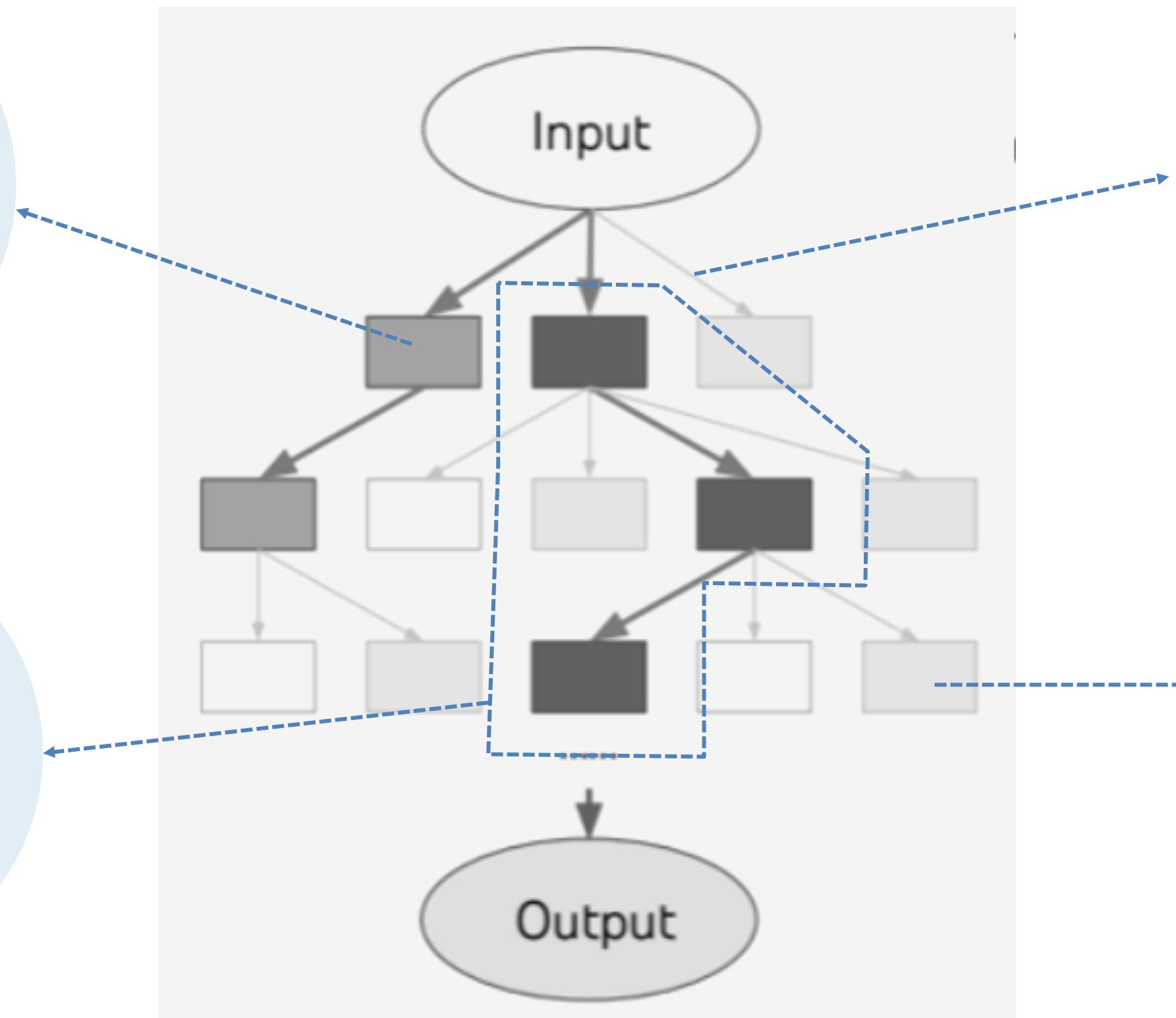
潛在可能

#故事線

④ 剪枝

果斷捨棄

#刪去法



從線性推理到策略探索的思維革命

分解

複雜問題 → 多個步驟

生成

多個下一步「思維」 = 樹的分支

評估

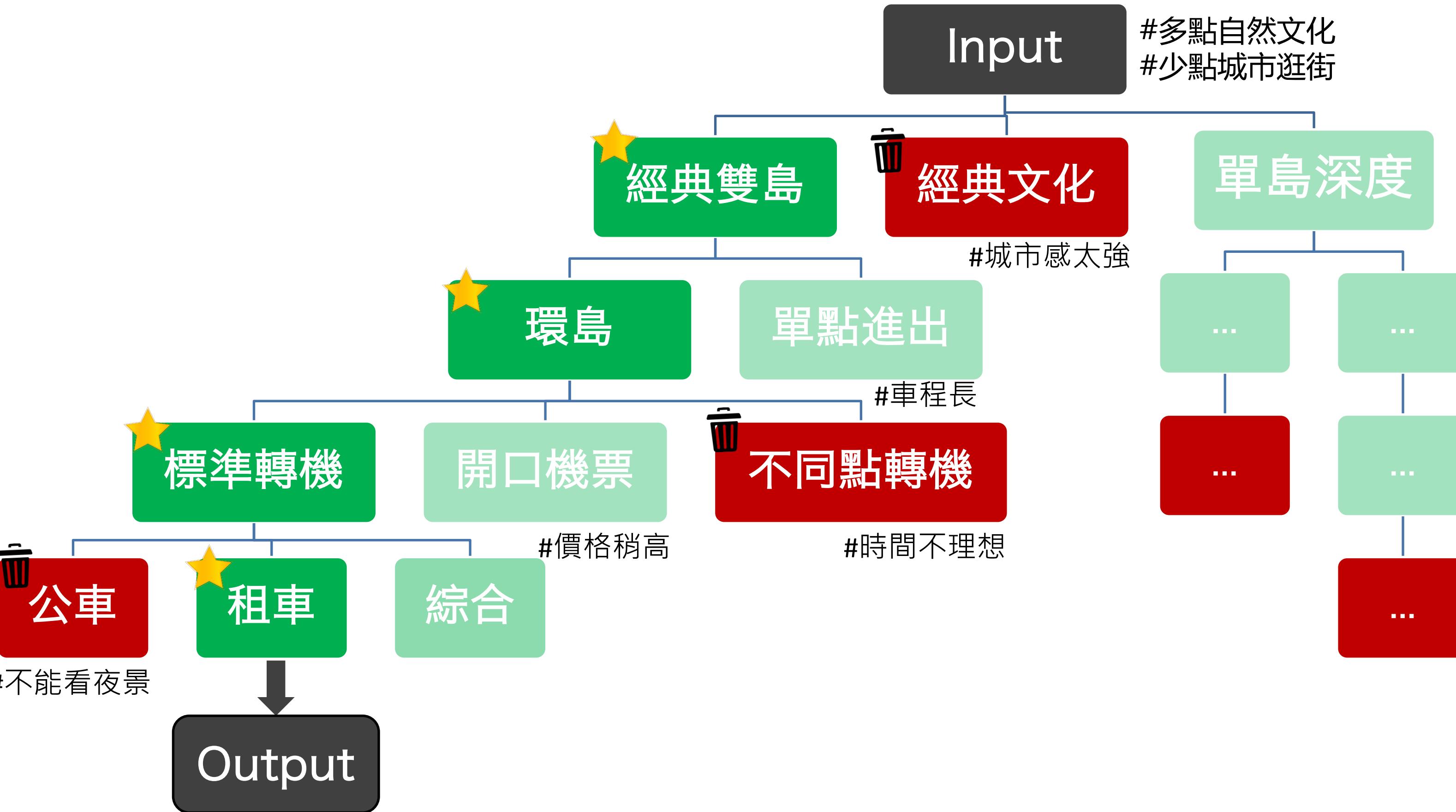
判斷每個**價值**或**可行性**
修剪沒希望分支，**聚焦有潛力路徑**

搜尋

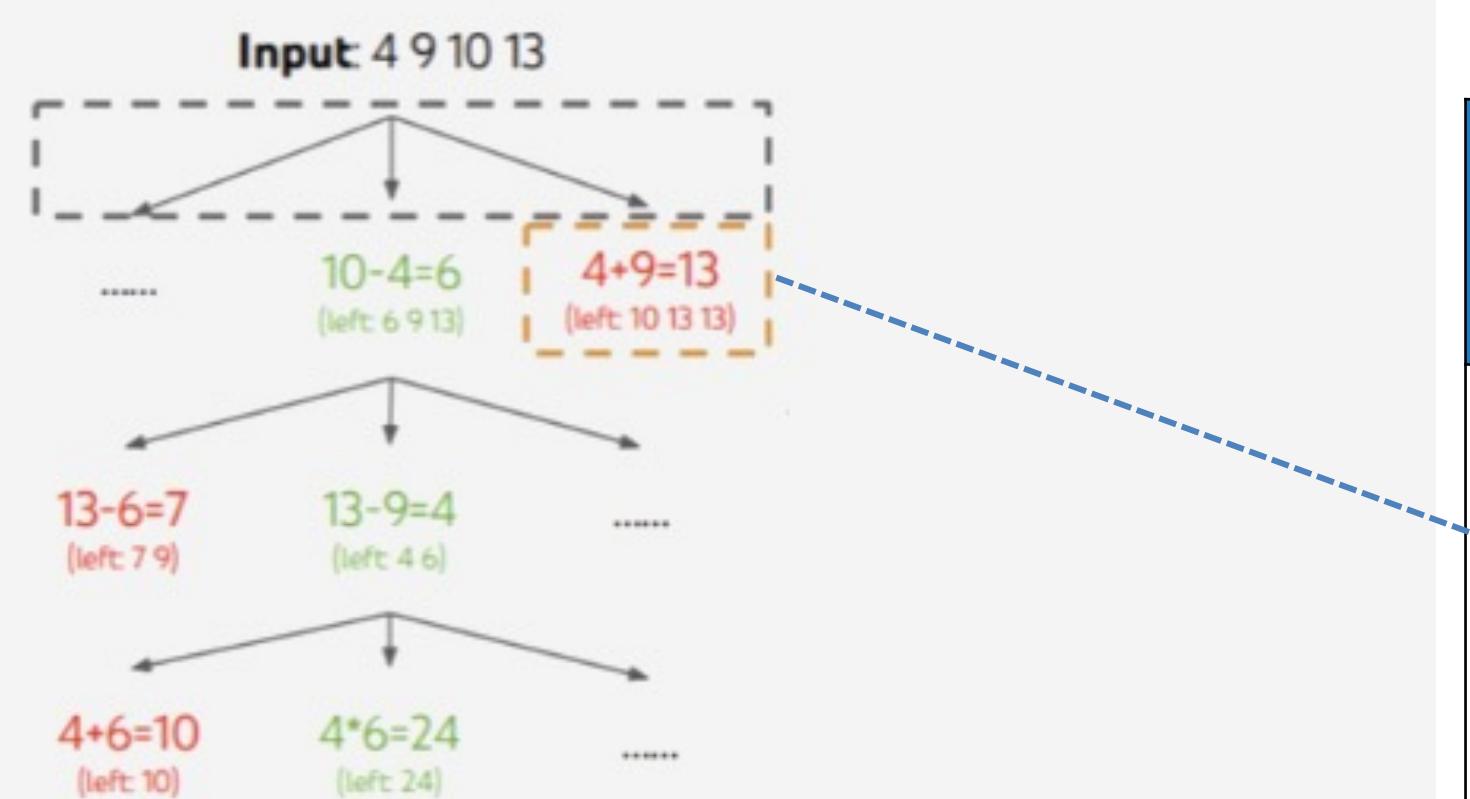
分數最高路徑 → 最終解決方案
廣度優先搜索 (BFS)、深度優先搜索 (DFS)

ToT 規劃最適合需求的夏威夷旅行

LLM Agent Planning

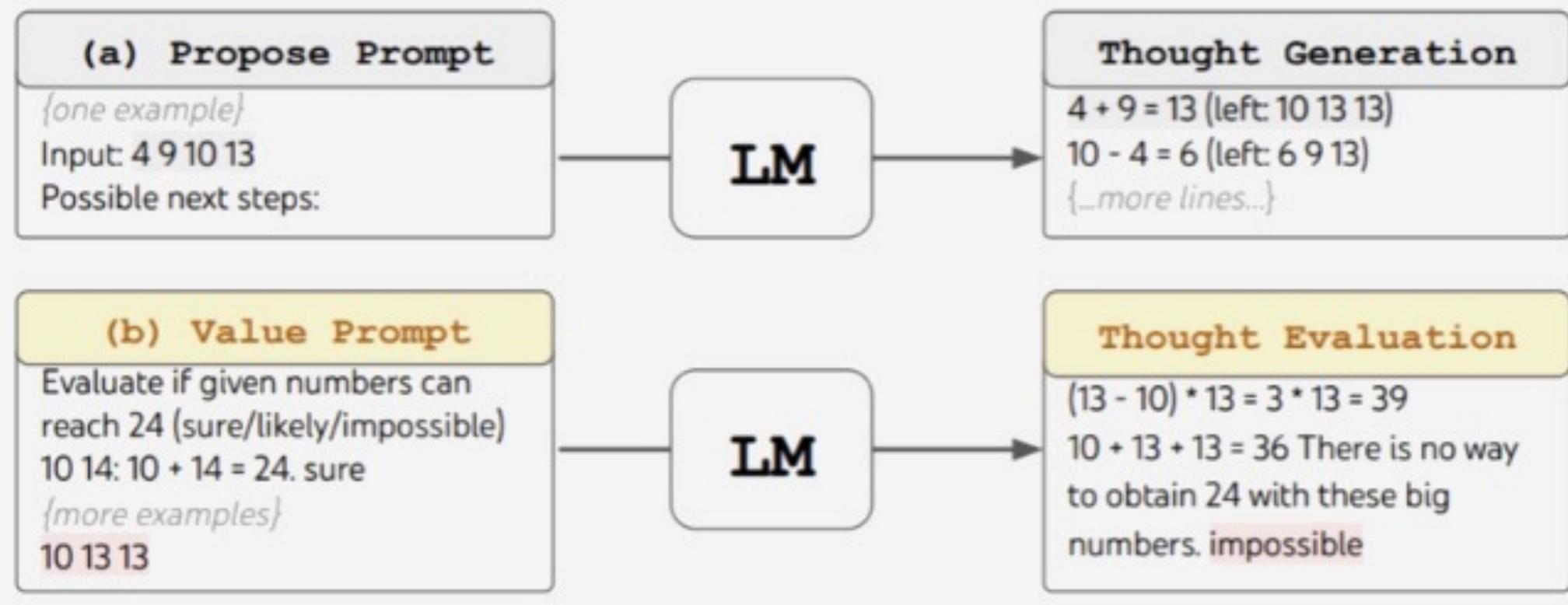


應用：24點遊戲如何評估錯誤節點



{4, 9, 10, 13}加減乘除得24

步驟1：分解生成
選擇“4+9=13”
變成{13, 10, 13}



步驟2：評估
{13, 10, 13}是否有可能等於24
答案是不可能

步驟3：剪枝
節點在這裡停止，並回朔上一步

應用：嘗試其他排列組合，找到最佳解決方案

步驟1：分解生成

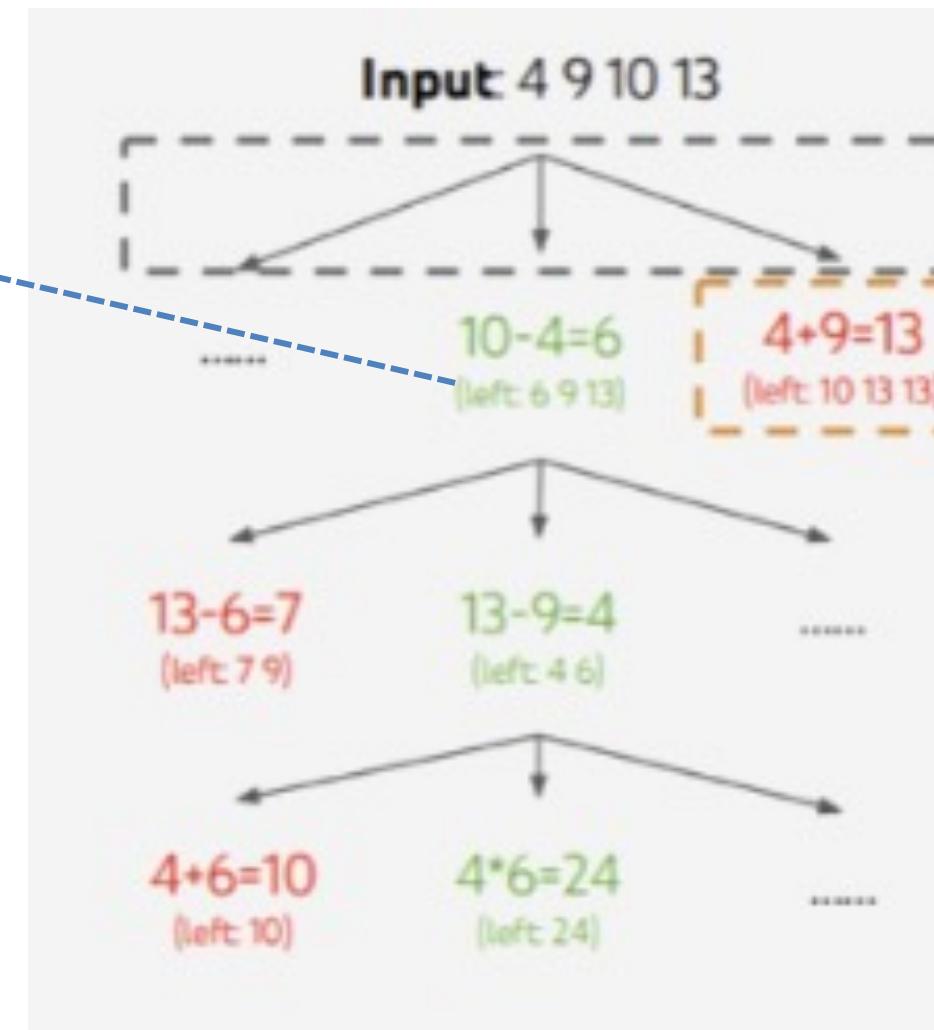
選擇“ $10-4=6$ ”
變成{6, 9, 13}

步驟2：評估

{6, 9, 13}是否有可能等於24
答案是可能

因為“ $13-9=4$ ”

剩下{4, 6}評估 $4*6=24$



| Method | Success |
|--------------------|---------|
| IO prompt | 7.3% |
| CoT prompt | 4.0% |
| CoT-SC (k=100) | 9.0% |
| ToT (ours) (b=1) | 45% |
| ToT (ours) (b=5) | 74% |
| IO + Refine (k=10) | 27% |
| IO (best of 100) | 33% |
| CoT (best of 100) | 49% |

Table 2: Game of 24 Results.

提升18倍 成功率

ToT像人類一樣深思熟慮的優劣勢

優勢

真正抓住你需要的「感覺」

無論A、B、C行程，都是一個主題完整、氛圍統一的深度之旅
而不是景點的拼湊，它提供是一種體驗，而不只是一個清單

過程更穩健、更高品質決策

想過很多可能性，所以能像個專家告訴你該如何選擇
並從多個候選方案中選出最優解

劣勢

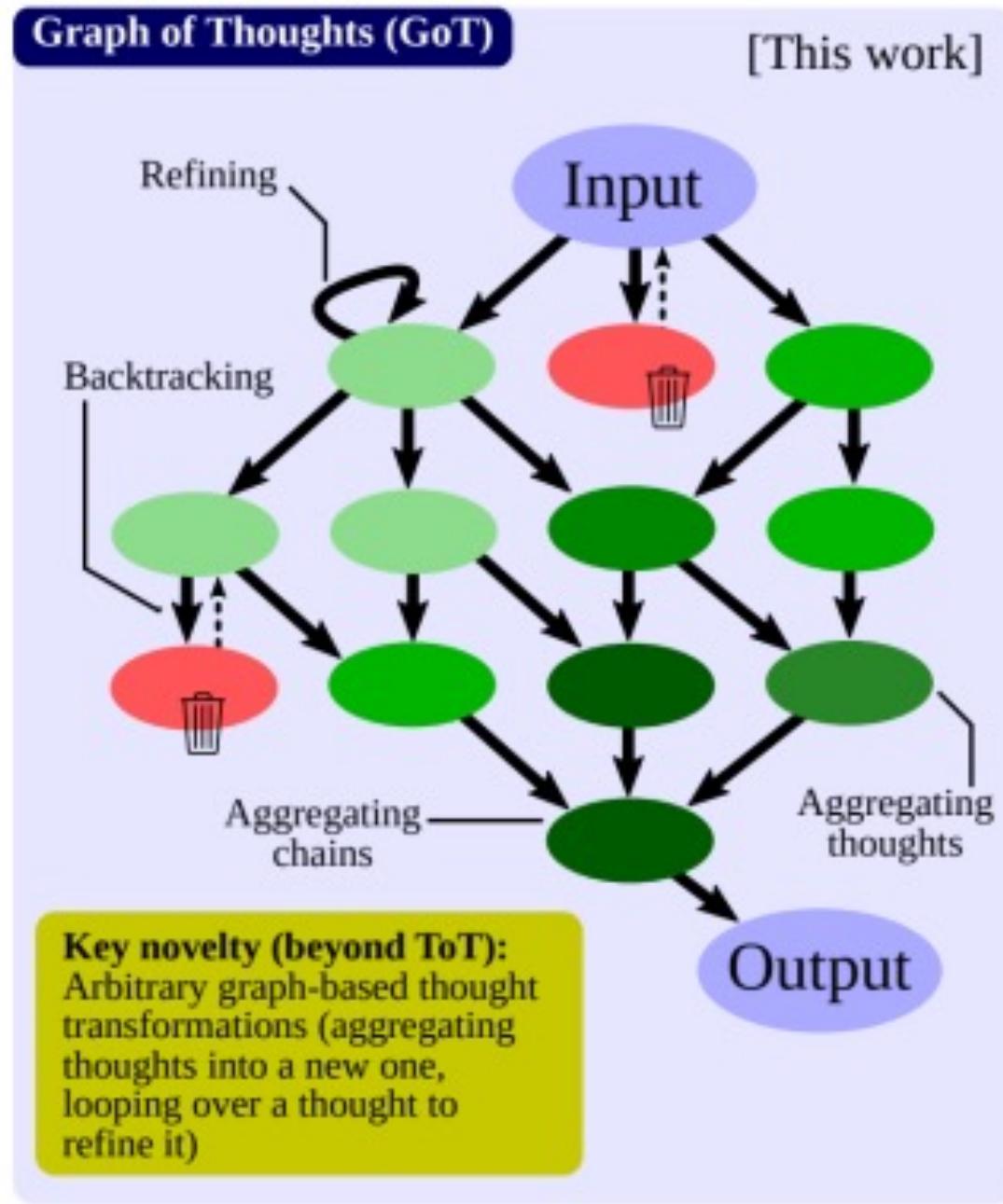
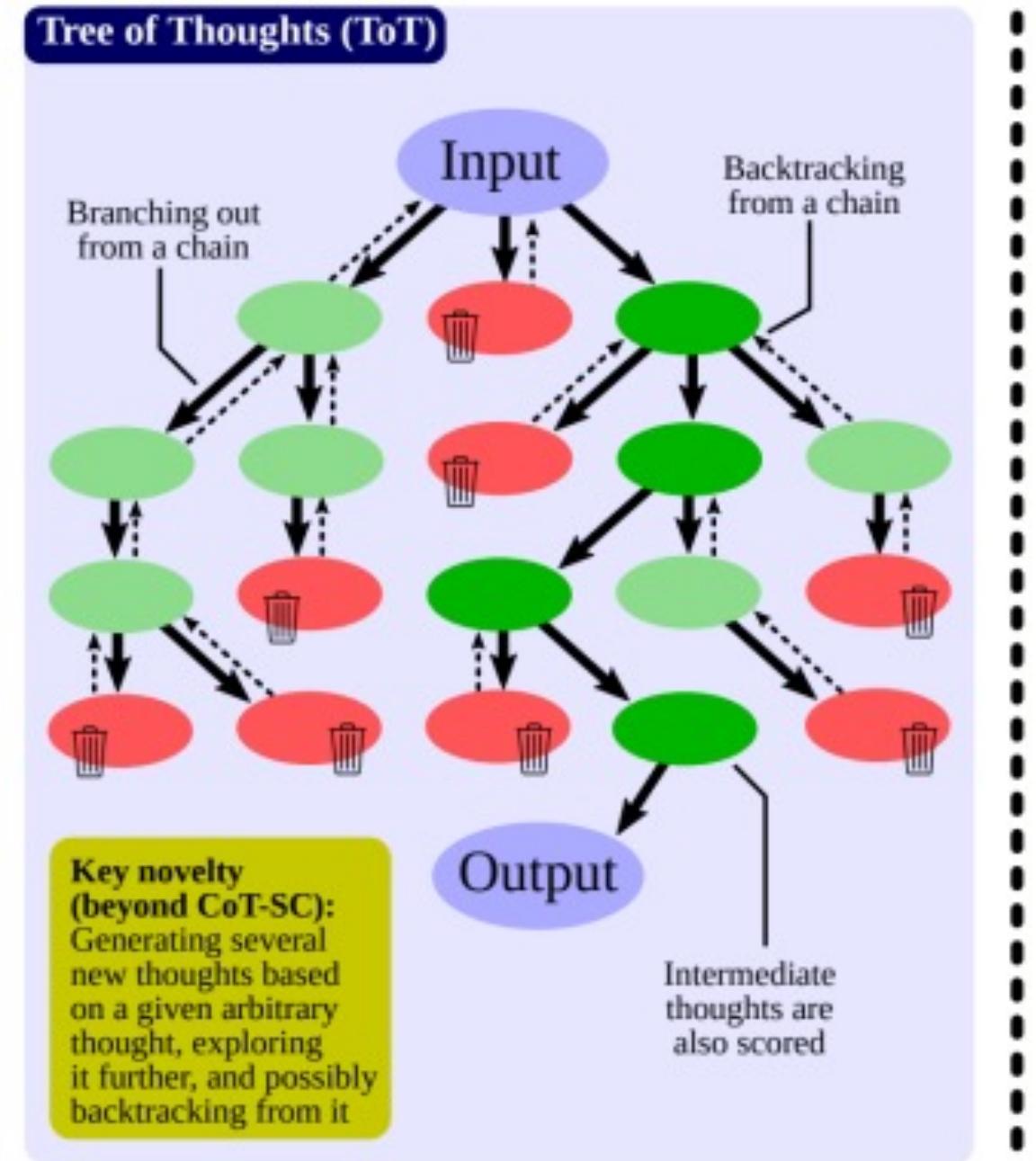
耗時且昂貴

經過腦力激盪，需要等待更久，付出更高的成本

可能想太多

有時為了規劃「最完美」旅程，設計5點起床、開車3小時去秘境
雖然看起來很棒，但對一般人來說太累了

但，如果樹枝能重新交織呢？
為什麼一定要選一個？不能全都要嗎？



思維圖 GoT Graph of Thoughts

想法可合併、循環*
思考結構更靈活
綜合不同觀點

不是在多個路線中選一個
而是透過融合不同路線優點
創造全新、更適合路線

*循環：帶著新的資訊或限制，重新回到之前的某個思考節點，進行迭代

04

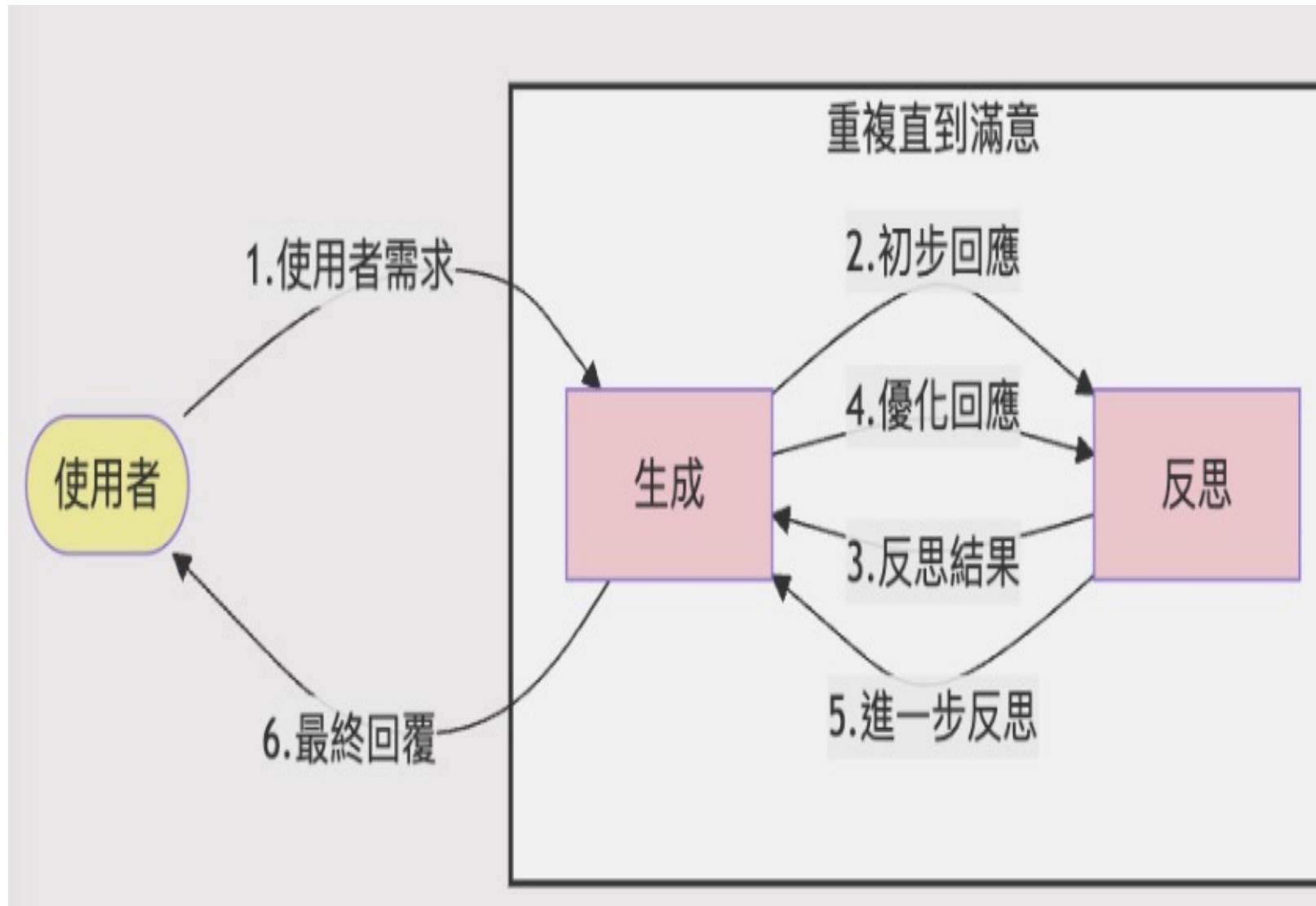
Reflection & Refinement

反思與改進機制

- 隨著 LLM 被應用在更複雜的 Agent 系統中，光靠單次輸出已無法滿足任務需求。
- 需要引入「反思」與「優化」來強化決策與內容品質。

| 類型 | Self-Refine | Reflexion |
|----|----------------------|-------------------------------|
| 特色 | 對輸出內容進行評估與正，反覆優化提升品質 | 對過去任務進行反思，產出經驗記憶，遇到類似情境時可輔助行動 |

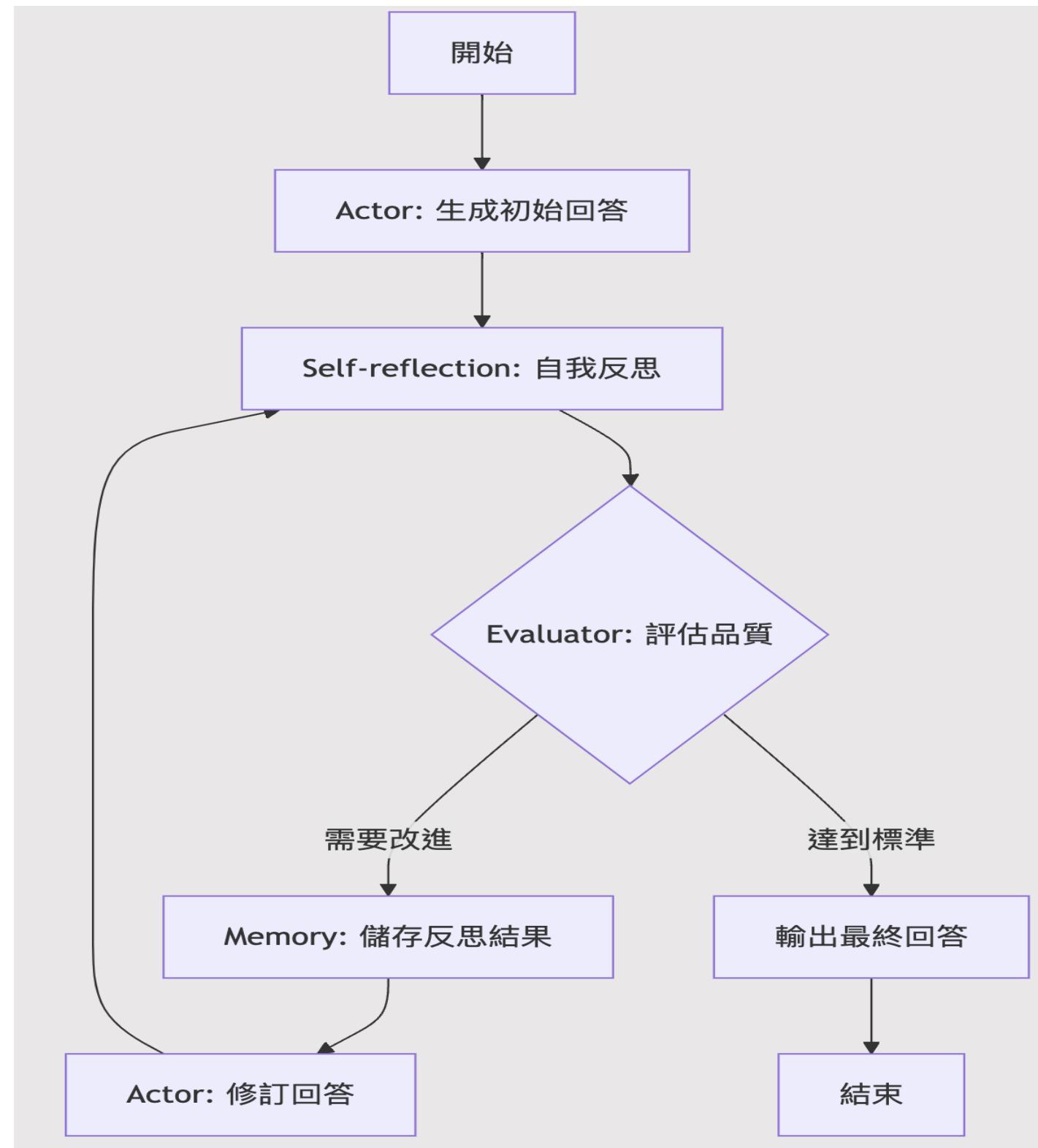
Self-Refine 流程



類似寫作時的潤稿修改

1. 初次生成答案
2. 自我評估
3. 根據檢討進行修改
4. 重複進行 (可設定迭代次數)

Reflexion流程



類似經驗筆記

1. 初次生成答案
2. 觸發反思 → 模型描述失敗原因
3. 得到結果 (成功/失敗)
4. 儲存反思到 Memory
5. 下次任務前讀取記憶 → 調整策略

情境

第一次任務輸出（初版行程）

- 早上去珍珠港
- 下午去 Hanauma Bay 浮潛
- 晚上吃飯店 buffet



- ✖ 行程時間衝突（浮潛與珍珠港距離遠）
- ✖ 體力負擔大
- ✖ 沒考慮交通

Self-Refine 的處理方式

Self-Refine可做的修改

- 把珍珠港和浮潛改成不同天
- 調整活動順序
- 加上一點說明如「建議提早出發」



- ! 無法記住失敗，可能會再犯錯
- ! 沒有真正學會合理安排

Reflexion 的處理方式

記錄反思內容

我過度集中安排在一天的活動，忽略地理距離與交通時間問題

存入記憶

避免當天行程超過 2 個，避免遠距移動排在同天

下次任務前調整

優先查詢活動地點之間距離與交通
確認交通方式是否可行



- ✓ 從錯誤中學習行程規劃原則，未來任務不再犯相同錯誤
- ✓ 更像一個有經驗的導遊，而非只是會修文字的秘書

Reflexion 的缺點與限制

- 記憶如果「學錯東西」，會讓 Agent 長期陷入壞習慣
- 記憶管理成本高
- 當下可控性不如Self-Refine，不適合用於精準內容修正任務



LLM核心Planning組合：選對方法，解決問題

CoT

#熱心旅遊部落客

處理簡單、明確問題，需要快速、直接答案

「如何從檀香山機場開車到威基基海灘？」

ToT

#細心規劃控

處理開放式、多變數的複雜問題，目標是找到最佳解

「五天預算三萬，想極限運動和在地文化，最好行程組合？」

ReAct

#行動派實踐家

計畫成功與否，取決於即時、變動的外部資訊

「馬上查一下！預約餐廳還有沒有位子？」

Reflexion

#經驗豐富學長

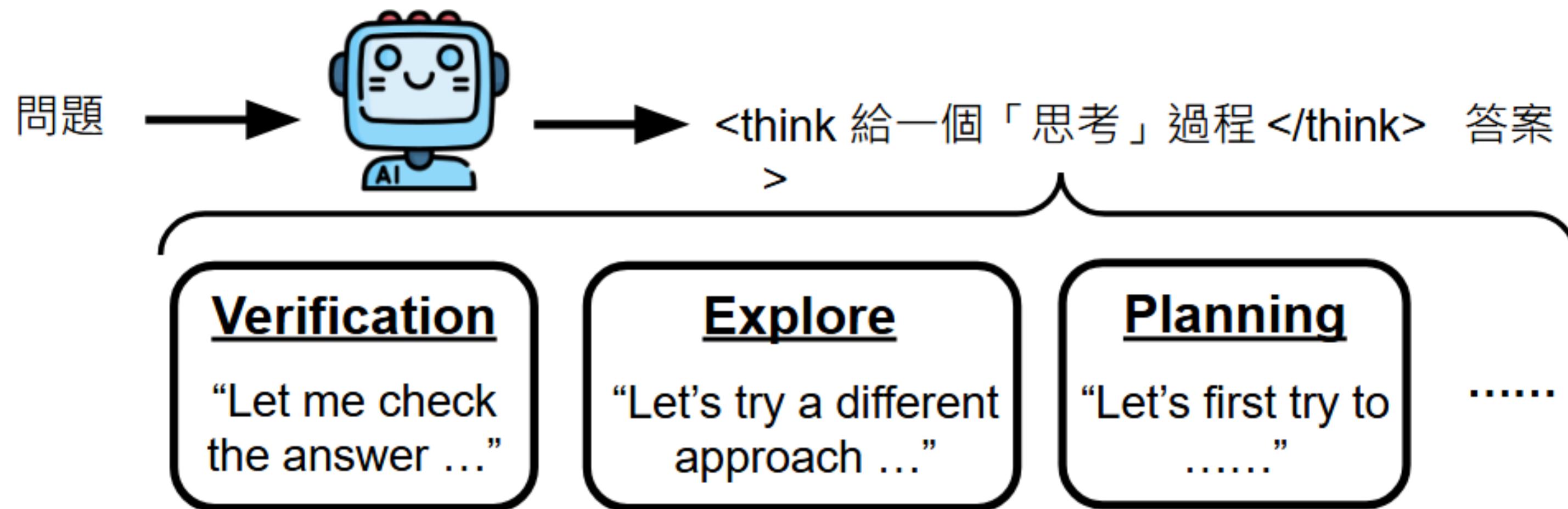
想避免重蹈覆轍，或讓已有計畫更周全、更完美

「這次行程規劃，有沒有什麼玩法是達人才知道的？」

05

Reasoning

深度思考



「推理」(Reasoning) (「Inference」字面翻譯類似，但意思完全不同)

**Test-Time
Compute**

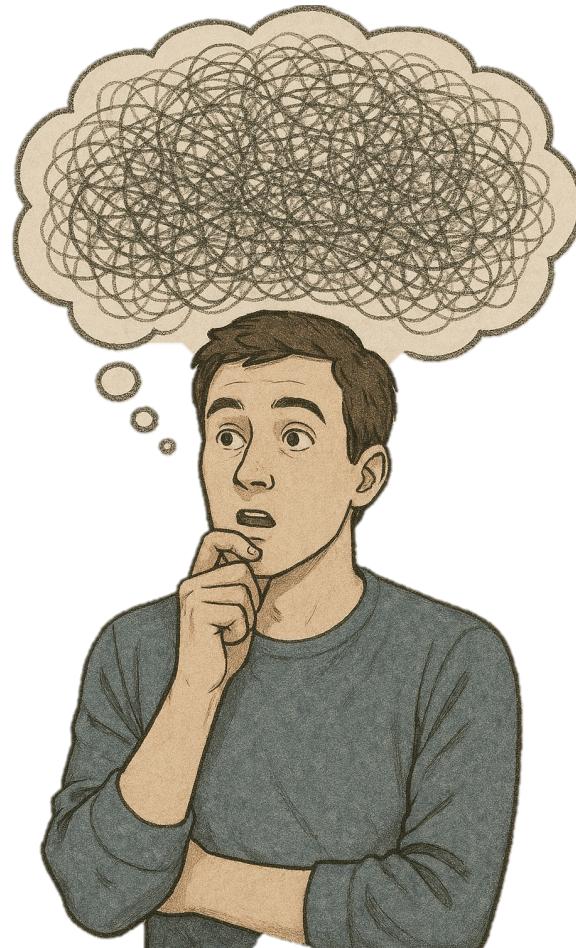
第一堂課：“深度不夠，長度來湊”

「深」到底是什麼？

想比較久？



內容比較多？



「深」到底是什麼？

層次更高
(Hierarchical)

邏輯更縝密
(Structured)

深度思考

一般思考

會分層進行，例如先拆解問題 → 探索可能性 → 評估可行性 → 整合與決策。這種結構性的思考像是在建造一棟推理的建築。

強調因果關係、邏輯步驟、每個推論之間有清楚連接，不只是對、還要「為什麼對」。

只進行一次性、線性的聯想或判斷，根據已知資料快速產生反應。

「直覺式」的，沒明確邏輯鏈，像是直接根據記憶回答。

「深」到底是什麼？

思考有回饋機制
(Reflective & Iterative)

能處理模糊與多解問題
(Open-ended Reasoning)

深度思考

一般思考

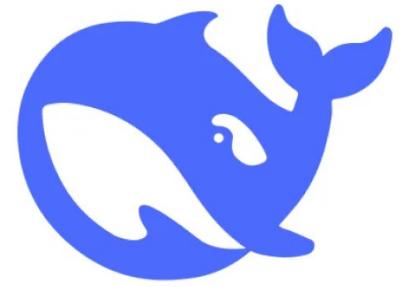
包含**自我檢查、反思與修正**，思考過程可以重複進行、逐步優化。

能處理**需要評估、比較、想像的開放式問題**，如「請設計一個方案」「請預測可能結果」，需要更多層面參與。

做完就做完了。

只處理明確問題，如選擇題或事實查詢。

具備「深度思考」能力的模型或產品



Deepseek R1

檀香山機場



Anthropic



OpenAI
ChatGPT **4.0**

一場精彩旅程
不在於都照著計畫

而在於能
拆解複雜問題
選擇最適方案

並在每次抉擇中…
勇於重新思考方向

