

Prompt Specificity in River-Crossing Logic: A Comparative Study of General-Purpose vs. Reasoning-Optimized LLMs

Axton Liu
Independent AI Prompt Engineering Researcher
hey@axtonliu.ai
<https://axtonliu.ai>

March 26, 2025

Abstract

This paper investigates how different prompting strategies affect large language models (LLMs) when solving a seemingly straightforward puzzle: transporting 7 people across a river in a 2-person boat. We compare “general-purpose” models (GPT-4o, GPT-4.5, Claude 3.5 Sonnet, Gemini 2.0 Flash, DeepSeek V3, Grok 3) and “reasoning-optimized” models (GPT o1, GPT o3-mini-high, Claude 3.7 Sonnet Extended Thinking, Gemini 2.5 Pro, Gemini 2.0 Flash Thinking, DeepSeek R1, Grok 3 Thinking) under three prompt conditions:

1. No Chain-of-Thought (CoT), 2. A simple CoT hint (“Think step by step”), 3. A more detailed CoT prompt (“Think step by step and write the detail analysis”).

We observe that some general-purpose models fail unless given a very explicit prompt—sometimes contradicting expectations of their advanced reasoning abilities—whereas others (including newly updated “general” versions) succeed even without CoT. Notably, GPT-4o, an ostensibly reliable GPT-4 variant, consistently fails under all prompt types. Meanwhile, all tested reasoning-optimized models solve the puzzle correctly in every scenario. We discuss how these findings reveal surprising vulnerabilities in older or baseline models, question whether prompt hints merely trigger existing logic modules, and highlight how “general vs. reasoning” boundaries are increasingly blurred.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in tasks requiring knowledge retrieval, text generation, and limited forms of logical reasoning. However, **small yet precise** logic puzzles—like the classic “boat can carry two, 7 people crossing” problem—can expose unexpected weaknesses. One particularly striking example is the consistent failure of **GPT-4o**, despite GPT-4 being widely considered “relatively reliable” in logical tasks. This highlights the reality that **some older or baseline LLM versions still struggle with seemingly trivial puzzles**, even though more advanced or newly fine-tuned models handle them with ease.

To explore *why* such failures occur, we investigate:

1. How do standard vs. detailed Chain-of-Thought (CoT) prompts affect puzzle-solving accuracy in LLMs?
2. Why do certain older variants (e.g., GPT-4o) fail systematically, while many reasoning-optimized or newly updated “general” models succeed?
3. Does “Think step by step” truly switch on an internal logic module, or is it insufficient for some LLMs without an even more elaborate directive?

By comparing both “general-purpose” and “reasoning-optimized” models under three prompt conditions (no CoT, simple CoT, and detailed CoT), we shed light on the interplay of prompt specificity and model

architecture. In doing so, we illustrate a potentially **”shocking” fact: some older or established models can repeatedly fail a basic puzzle**—a cautionary reminder that vendor or version labels do not guarantee robust reasoning for every scenario.

2 Related Work

Chain-of-Thought prompting (CoT) has been shown to improve multi-step reasoning by prompting the model to ”show its work” [1, 2]. Meanwhile, commercial LLM deployments often introduce specialized ”reasoning” or ”extended thinking” variants, further complicating the landscape [3, 4]. Although prior research highlights that the same model can perform significantly better or worse under slightly different prompts [1, 2], less attention has been paid to **consistently failing** older or baseline versions that do not respond well to any form of CoT.

Our study contributes by:

- Demonstrating that certain well-known older variants (GPT-4o) can exhibit surprising failures,
- Systematically comparing ”general” vs. ”reasoning” solutions on a single puzzle with minimal vs. detailed CoT,
- Discussing how the puzzle’s language ambiguity (”来回几次?”) influences numeric answers, further stressing the importance of *clear puzzle definitions*.

3 Methodology

3.1 Puzzle Description

We use the puzzle:

”A boat can carry exactly 2 people at a time. There are 7 people who need to cross a river. How many trips back and forth does it take to get everyone across?”

Key constraints:

1. The boat cannot move by itself; at least 1 person must row.
2. A minimal solution typically yields **11** single crossings (or 5 complete round trips + 1 final one-way).
3. The Chinese phrasing ”来回几次” can cause confusion, producing answers from 5 to 13.

3.2 Prompt Types

Each model was tested under **three prompt conditions**:

1. **No CoT** — No special instructions, just the puzzle question.
2. **Simple CoT** — Appended ”Think step by step.”
3. **Detailed CoT** — Appended ”Think step by step and write the detail analysis.”

3.3 Models: General vs. Reasoning

We group models into:

- **General-Purpose:** GPT-4o, GPT-4.5, Claude 3.5 Sonnet, Gemini 2.0 Flash, DeepSeek V3, Grok 3.
- **Reasoning-Optimized:** GPT o1, GPT o3-mini-high, Claude 3.7 Sonnet Extended Thinking, Gemini 2.5 Pro, Gemini 2.0 Flash Thinking, DeepSeek R1, Grok 3 Thinking.

Notable Surprise: GPT-4o, historically aligned with GPT-4’s robust reputation, performed poorly here. Meanwhile, some ”general” versions (DeepSeek V3) show strong built-in logic, emphasizing that these category labels are no longer reliable proxies for actual performance.

3.4 Data Collection

- **Test Setup:** Each model was queried in an isolated session, avoiding prior context contamination.
- **Evaluation:** We labeled an answer "Correct" if it either yields the minimal crossing count (11 single trips) or clearly explains the 5–6 round-trip logic.
- **Repeated Trials:** If a model gave inconsistent answers across attempts, we recorded the first stable or majority result.

4 Results

Table 1: Performance Across Prompts

Model	Type	No CoT	Simple CoT	Detailed CoT	Comment
GPT-4o	General	Fail	Fail	Fail	Unexpectedly fails in all conditions—see Sec. 5.1
GPT-4.5	General	Fail	Fail	Correct	Needs highly explicit "detailed analysis"
Claude 3.5 Sonnet	General	Fail	Fail	Correct	Similar pattern to GPT-4.5
Gemini 2.0 Flash	General	Fail	Fail	Fail	All tested prompts yield incorrect answers
DeepSeek V3	General	Correct	Correct	Correct	Built-in stepwise logic, no special CoT needed
Grok 3	General	Correct	Correct	Correct	Also robust without explicit CoT
GPT o1	Reasoning	Correct	Not tested	Not tested	Provided direct formula ($2n-3 = 11$)
GPT o3-mini-high	Reasoning	Correct	Not tested	Not tested	Also used the minimal crossing formula
Claude 3.7 Ext Thinking	Reasoning	Correct	Not tested	Not tested	Detailed logic chain yields 11
Gemini 2.5 Pro	Reasoning	Correct	Not tested	Not tested	Enumerates 11 one-way trips consistently
Gemini 2.0 Flash Thinking	Reasoning	Correct	Not tested	Not tested	Some numeric confusion but final answer correct
DeepSeek R1	Reasoning	Correct	Not tested	Not tested	Clarifies 5 round-trips vs. 11 single-crossings
Grok 3 Thinking	Reasoning	Correct	Not tested	Not tested	Step-by-step approach, final 11

5 Discussion

5.1 GPT-4o’s Persistent Failure

Given GPT-4’s reputation for advanced reasoning, one might expect GPT-4o—an older or baseline variant—to at least succeed under a detailed CoT prompt. Surprisingly, it repeatedly fails, often providing “9” or “13” as final answers. Possible factors include:

- **Incomplete version or training data:** GPT-4o may not incorporate the same updates/fine-tuning as GPT-4.5.
- **Semantic confusion in “来回几次”:** GPT-4o may systematically misinterpret “round trip” vs. single crossing.
- **Fixed flawed internal heuristic:** It might rely on a baked-in formula or short-circuited logic not easily overridden by user prompts.

This phenomenon challenges the assumption that “triggering a logic chain” (via CoT) is always enough; if the model’s internal representation or heuristic is off, **even explicit step-by-step instructions might not correct it.**

5.2 Prompt Specificity vs. Internal Logic

The fact that GPT-4.5 and Claude 3.5 remain incorrect under “Think step by step” yet become correct with “Think step by step *and write the detail analysis*” raises questions:

- Is a short CoT hint insufficient to override default shortcuts?
- Does a more elaborate prompt forcibly “uncork” a deeper reasoning process?

Such outcomes indicate that prompt engineering is not merely about enabling any chain of thought, but about ensuring the *model* invests in verifying each step. Meanwhile, updated “general” models like DeepSeek V3 or Grok 3 and all “reasoning-optimized” models appear to do so automatically.

5.3 Some Older Models Consistently Fail on Simple Tasks

Our findings highlight a broader reality in the LLM ecosystem:

“Some established or older LLM variants still fail at seemingly trivial logic.”

This has major implications:

1. **User Trust:** Non-technical users might assume a big-name model is infallible; these results show that baseline or older versions can disappoint on tasks requiring precise counting.
2. **Upgrade Necessity:** Vendors must continually patch or re-fine-tune older releases to keep them competitive, especially on logic tasks that appear “too simple” to remain broken.
3. **Research & Testing:** Even “classic” problems can be revealing. The mismatch between marketing and actual capability underscores why thorough tests on simple puzzles still matter.

5.4 Language Ambiguity and Multiple Answers

Many answers included “5, 6,” or “9,” sometimes referencing the difference between a full round trip and single crossings. The puzzle’s wording “来回几次” can be read as “how many round trips?” or “how many one-way trips total?” Models that enumerated every crossing typically arrived at 11, while others only counted complete back-and-forth cycles. This discrepancy is further complicated by the final crossing not needing a return trip.

Hence, puzzle specification is crucial: If the question clarifies “count each single crossing,” or “count only how many times the boat leaves side A,” ambiguity is minimized.

6 Conclusion and Future Work

We presented a comparative study on a classic 2-person boat puzzle across multiple LLMs and prompt strategies. Key observations:

1. **Older or baseline models (GPT-4o) can shockingly fail** even under explicit CoT prompts. This calls into question the assumption that "older GPT-4 is automatically strong in logic."
2. **Prompt specificity** is highly influential. Some LLMs need fully spelled-out instructions ("write the detail analysis") to correct prior mistakes.
3. **Reasoning-optimized** models prove robust in all tested conditions, but certain updated "general" models also exhibit strong "implicit" CoT.
4. **Language ambiguity** significantly impacts numeric answers, underscoring the importance of puzzle clarity.

Future directions include:

- **Expanding to additional micro-puzzles** (e.g., "Wolf-Goat-Cabbage," "Missionaries and Cannibals") and testing how older vs. updated versions of popular LLMs fare.
- **Investigating internal heuristics** behind an LLM's repeated miscalculation—could it be overwritten with dynamic CoT instructions or require further fine-tuning?
- **Large-scale prompt sensitivity studies** focusing on multiple languages and more nuanced instructions to see how LLMs adapt or fail.

We hope our findings serve as a **cautionary tale**: even "big names" can falter on "simple" tasks, and subtle changes in prompt wording can be the difference between a correct solution and a baffling error.

References

- [1] Wei, J. et al. (2022). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*.
- [2] Kojima, T. et al. (2022). *Large Language Models are Zero-Shot Reasoners*.
- [3] Anthropic, OpenAI, DeepMind, etc. (2023–2025). *Public Model Release Notes*.
- [4] Nye, M. et al. (2022). *Show Your Work: Scratchpads for Intermediate Computation with Language Models*.

A Appendix: Sample Outputs

The screenshots listed in this appendix showcase examples of outputs from different models under three prompting strategies: **No CoT**, **Simple CoT**, and **Detailed CoT**. Unless otherwise specified, all images are taken from actual conversation results.

A.1 GPT-4o

A.1.1 No CoT



Figure 1: GPT-4o under *No CoT* prompt. Example answer (incorrect).

A.1.2 Simple CoT

ChatGPT 4o

Share

一只小船能载 2 人，一共有 7 人要过河，问要来回几次才能把所有人都送到对岸？ Think Step by Step

这是一个经典的“渡河问题”，我们一步一步来分析。

问题条件：

- 小船 每次最多载 2 人；
- 一共有 7 个人；
- 船 必须有人划回来，也就是说不能自动回船；
- 目标是 所有人都到对岸；
- 问：最少需要来回几次？

解题思路：

由于船每次最多载 2 人，并且必须有人将船划回来，我们要尽量减少“回程”的次数，并且让回程尽量由一个人来完成，以节省资源。

我们用下面的方式来思考这个问题：

第一步：送人过去（最有效率的方式）

每次送 2 个人过去，1 个人回来，相当于：

- 每次净增加 1 个人在对岸。

比如：

- A 和 B 过去，A 回来：（对岸 1 人）
- C 和 D 过去，C 回来：（对岸 2 人）
- E 和 F 过去，E 回来：（对岸 3 人）
- G 和 A 过去（最后一趟，不需要回来）。（对岸 5 人）

等一下，感觉哪里不对？来看看 更合理的方式。

使用“来回计数法”：

我们假设一次“来回”指的是“从这边出发、再把船划回来”——不一定每次都需完整的来回，但我们可以从最少次数来估算。

核心策略：用船的有限资源最大化效率

- 两个快递员（船夫）原则：
 - 一种经典方法是：挑出两个“船夫”（比如 A 和 B），他们负责大部分的送人和回程。
 - 其他人通过这两个人的“运送”逐步送到对岸。

步骤详细分析：

我们挑出 A 和 B 作为“运送者”，其余 5 人依次送过去。

过程：

- A 和 B 过去（对岸：A、B），B 回来（剩下 A）
- C 和 D 过去（对岸：A、B、C、D），A 回来（剩下 B）
- E 和 F 过去（对岸：B、C、D、E、F），B 回来（剩下 A）
- G 和 A 过去（对岸：C、D、E、F、G、A、B）🏁 结束！

总计次数分析：

每一组的运送过程需要：

- 送 2 人过去（1 次）
- 1 人回来（1 次）

我们这样一共做了 3 组运送（C&D、E&F、G&A）+ 最开始 A&B 先过去：

- 第一次：A&B 过去，B 回来（2 次）
- 第二组：C&D 过去，A 回来（2 次）
- 第三组：E&F 过去，B 回来（2 次）
- 第四组：G&A 过去，不用回来（1 次）

总共：2 + 2 + 2 + 1 = 7 次 🟢

8

A.1.3 Detailed CoT

ChatGPT 4o

Share

一只小船能载 2 人，一共有 7 人要过河。问要来回几次才能把所有人都送到对岸？ Think step by step and write the detail analysis

我们来一步一步分析这道经典的“渡河问题”：

🧠 题目重述：

- 一只小船能载 2 人
- 一共 7 人 要过河
- 问：最少需要来回几次 才能把所有人都送到对岸

🧩 解题策略：

由于小船只能坐 2 人，且船必须有人划回来，因此我们要思考以下几个关键点：

- 每次送人过去的时候，可以送 1 人或 2 人
- 船要回来接人，必须 至少 1 人划回来
- 总体目标：让 7 人都到对岸，且船也可以留在最后一边

📝 假设：小船起初在出发岸（A 岸），我们记：

- A：出发岸（起点）
- B：对岸（终点）

✅ 解题过程（逐步演示）

我们使用最常见的策略：每次送 2 人过去，一个人划回来

第 1 轮：

- 两人从 A 划船到 B → 2 人到了 B ✅ (共 1 次)
- 一人从 B 划回到 A → 回到 A ✅ (共 2 次)

第 2 轮：

- 又送 2 人过去 → B 上共 4 人了 ✅ (共 3 次)
- 一人划回 A → 回到 A ✅ (共 4 次)

第 3 轮：

- 再送 2 人过去 → B 上共 6 人了 ✅ (共 5 次)
- 一人划回 A → 回到 A ✅ (共 6 次)

第 4 轮（最后）：

- 剩下最后 1 人和划船人一起过河 → 全部 7 人都到 B ✅ (共 7 次)

✅ 总结：

- 一共来回了 7 次
- 详细顺序如下：
 - A → B (送 2 人)
 - B → A (回 1 人)
 - A → B (送 2 人)
 - B → A (回 1 人)
 - A → B (送 2 人)
 - B → A (回 1 人)
 - A → B (最后 2 人)

📌 答案：

👉 最少需要 7 次来回，才能把所有 7 人全部送到对岸。

如果你对这个问题感兴趣，还可以尝试推广：

- 如果是 N 个人呢？
- 如果船只能坐 1 人？ 或者能坐 3 人？

要不要我帮你写一个通用的推导公式或动画演示？

10

A.2 GPT-4.5

A.2.1 No CoT



Figure 4: GPT-4.5 under *No CoT* prompt (incorrect).

A.2.2 Simple CoT

ChatGPT 4.5 Share

一只小船能载 2 人，一共有 7 人要过河。问要来回几次才能把所有人都送到对岸？ Think Step by Step

我们一步一步地分析这个问题：

步骤一：分析问题背景

- 一条船每次最多载 2 人。
- 现在需要把 7 个人送到河对岸。

步骤二：确定过河策略

- 每次从此岸到彼岸最多送 2 人，回来必须有人把船划回来（船不会自动返回）。
- 因此，每次渡河的策略是：
 - 第一次：2 人过去，1 人回来（送过去 1 人）。
 - 第二次：再送 2 人过去，1 人回来（再送过去 1 人）。
 - 重复这一过程，直到最后一次渡河后无需再返回。

步骤三：计算详细步骤

我们逐步模拟：

次数	去程（此岸→对岸）	回程（对岸→此岸）	已送达对岸的人数
1	去 2 人	回 1 人	1 人
2	去 2 人	回 1 人	2 人
3	去 2 人	回 1 人	3 人
4	去 2 人	回 1 人	4 人
5	去 2 人	回 1 人	5 人
6	去 2 人	回 1 人	6 人
7	去 2 人	不用再回来	7 人（全部送达）

步骤四：总结答案

- 总共需要 7 次去程 和 6 次回程。
- 因此，来回一共需要： $7 + 6 = 13$ 次。

✅ **最终答案：13次**

Figure 5: GPT-4.5 under *Simple CoT* prompt (still incorrect).

A.2.3 Detailed CoT

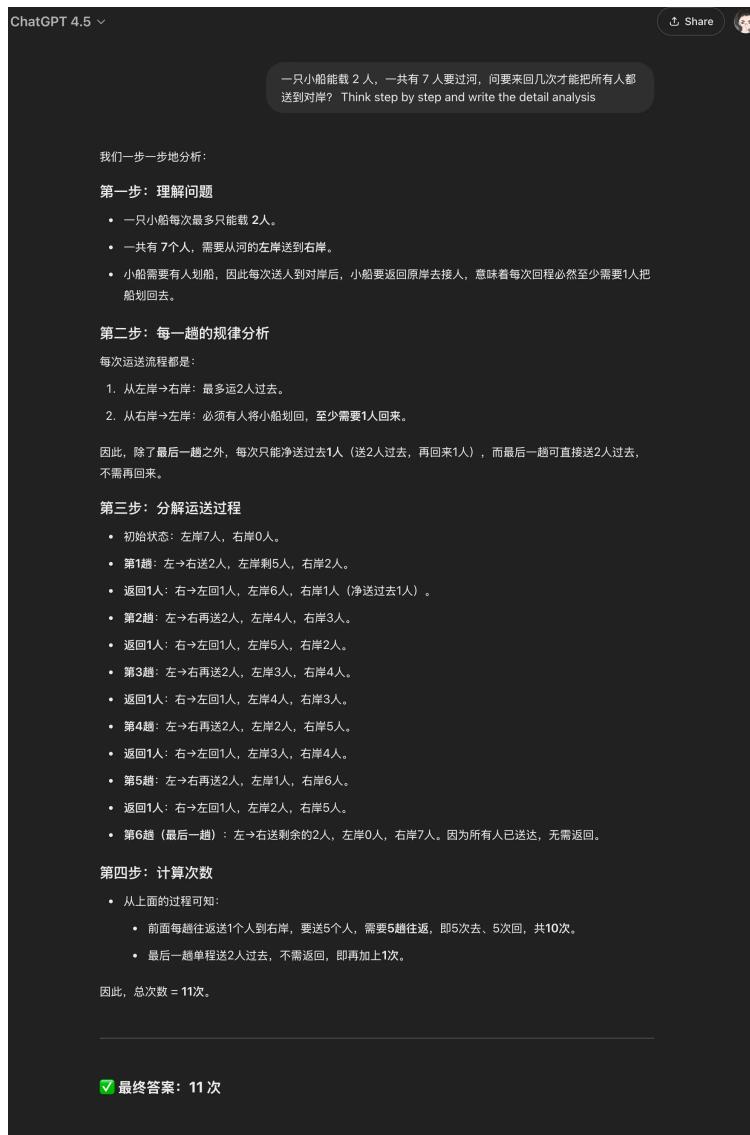


Figure 6: GPT-4.5 under *Detailed CoT* prompt (correct after detailed analysis).

A.3 Claude 3.5 Sonnet

A.3.1 No CoT



Figure 7: Claude 3.5 Sonnet with *No CoT* prompt (incorrect).

A.3.2 Simple CoT

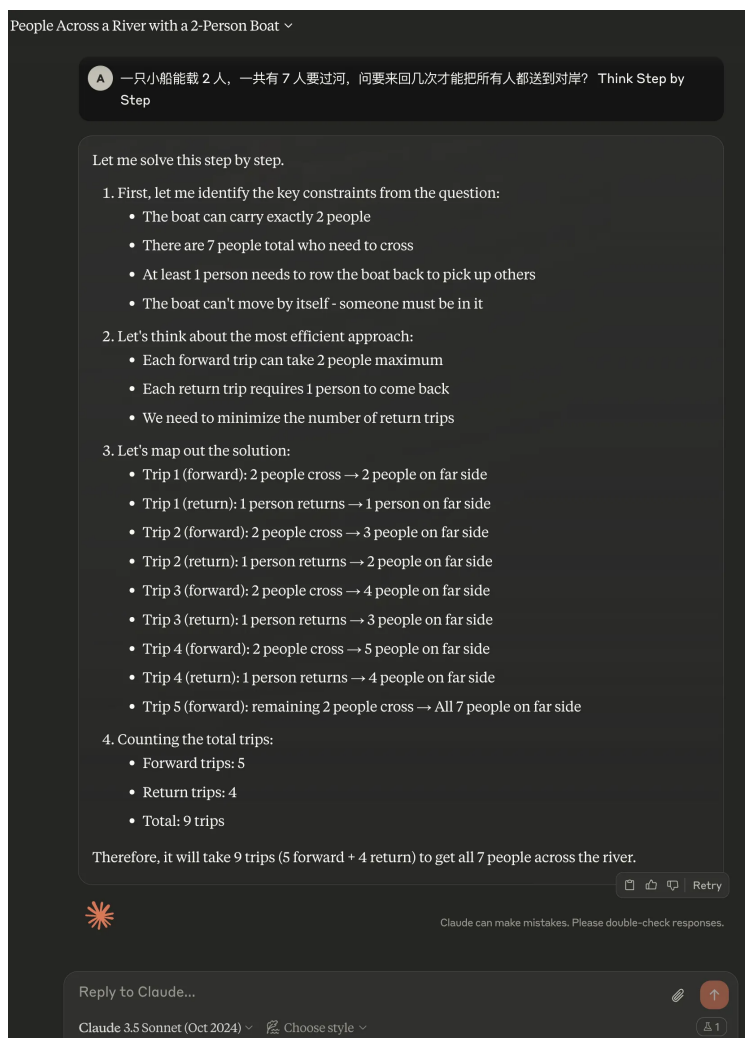


Figure 8: Claude 3.5 Sonnet under *Simple CoT* prompt.

A.3.3 Detailed CoT

A 一只小船能载 2 人，一共有 7 人要过河，问要来回几次才能把所有人都送到对岸？ Think step by step and write the detail analysis

Let me solve this river crossing puzzle step by step. First, let's clearly state the constraints from your message:

1. A boat can carry 2 people at most
2. There are 7 people total who need to cross
3. The boat needs someone to row it back (can't move on its own)
4. Everyone needs to reach the other side

Let's solve this systematically:

1. Initial setup:
 - Starting side: 7 people
 - Other side: 0 people
 - Boat capacity: 2 people
2. Let's track each crossing: Trip 1 (forward):
 - 2 people cross
 - Starting side: 5 people
 - Other side: 2 people

- Trip 3 (forward):
- 2 people cross
 - Starting side: 4 people
 - Other side: 3 people

- Trip 4 (return):
- 1 person returns
 - Starting side: 5 people
 - Other side: 2 people

- Trip 6 (return):
- 1 person returns
 - Starting side: 4 people
 - Other side: 3 people

- Trip 7 (forward):
- 2 people cross
 - Starting side: 2 people
 - Other side: 5 people

- Trip 8 (return):
- 1 person returns
 - Starting side: 3 people
 - Other side: 4 people

- Trip 9 (forward):
- 2 people cross
 - Starting side: 1 person
 - Other side: 6 people

- Trip 10 (return):
- 1 person returns
 - Starting side: 2 people
 - Other side: 5 people

- Trip 11 (forward):
- 2 people cross (final crossing)
 - Starting side: 0 people
 - Other side: 7 people

Final answer: It will take 11 trips to get all 7 people across the river.

The reason this is optimal:

- Each forward trip can take 2 people maximum
- Each return trip must have 1 person to row back
- To move 7 people, we need enough forward capacity (2 people/trip) to transport them all
- The trips alternate between forward (with 2) and return (with 1) until everyone is across

📄 📌 🔄 Retry



A.4 DeepSeek V3 (No CoT)

7人过河需5次来回

一只小船能载 2 人，一共有 7 人要过河，问要来回几次才能把所有人都送到对岸？

问题重述

题目说：“一只小船能载 2 人，一共有 7 人要过河，问要来回几次才能把所有人都送到对岸？”

理解题意

首先，我们需要明确几个关键信息：

1. 小船容量：每次最多可以载 2 人。
2. 总人数：7 人需要从河的一边（起始岸）运到另一边（对岸）。
3. 船的运作方式：
 - “来回一次”通常指的是船从起始岸到对岸，再返回起始岸的一个完整往返。
 - 每次运输都需要有人划船返回，除非所有人已经在对岸。

运输的基本模式

在过河问题中，通常需要有人将船划回起始岸以便继续运输其他人。如果船只能载 2 人，那么：

- 去程：起始岸到对岸，可以载 1 或 2 人。
- 回程：对岸返回起始岸，至少需要 1 人划船返回。

为了最大化运输效率，通常每次运输尽可能多的人到对岸，同时尽量减少返回的次数。因此，最优的策略可能是：

- 每次运 2 人到对岸，然后 1 人划船返回。

这样，每次“来回”可以净增加对岸的人数：

- 去: +2 人 (对岸), -2 人 (起始岸)
- 回: -1 人 (对岸), +1 人 (起始岸)
- 净效果: 对岸 +1 人, 起始岸 -1 人

计算运输步骤

让我们一步步计算：

初始状态:

- 起始岸：7 人
- 对岸：0 人
- 船在起始岸

运输过程：

1. 第一次运输:
- 运 2 人到对岸。
 - 起始岸: $7 - 2 = 5$ 人
 - 对岸: 2 人
 - 1 人划船返回。
 - 起始岸: $5 + 1 = 6$ 人
 - 对岸: $2 - 1 = 1$ 人
 - 来回次数: 1
2. 第二次运输:
- 运 2 人到对岸。
 - 起始岸: $6 - 2 = 4$ 人
 - 对岸: $1 + 2 = 3$ 人
 - 1 人划船返回。
 - 起始岸: $4 + 1 = 5$ 人
 - 对岸: $3 - 1 = 2$ 人
 - 来回次数: 2
3. 第三次运输:
- 运 2 人到对岸。
 - 起始岸: $5 - 2 = 3$ 人
 - 对岸: $2 + 2 = 4$ 人
 - 1 人划船返回。
 - 起始岸: $3 + 1 = 4$ 人
 - 对岸: $4 - 1 = 3$ 人
 - 来回次数: 3

A.5 Gemini 2.5 Pro (No CoT)



Figure 11: Gemini 2.5 Pro under *No CoT* prompt, enumerating the correct minimal crossing count.