CS5800 ALGORITHMS · AI/LLM PROBLEM

The Three-Jar Pouring Problem

PROBLEM

> Three jars hold integer quantities of water *A*, *B*, *C*. A **pour** operation picks two jars and transfers water from the larger to the smaller, exactly enough to double the smaller:
>
> $$\text{pour}(X, Y) \text{ with } X \geq Y \rightarrow (X - Y, 2Y)$$
>
> **Goal:** design a sequence of pours guaranteed to reach a state where at least two jars hold equal amounts, starting from any positive integers A, B, C.

HOW THIS WORKS

This problem is hard to solve from scratch in a reasonable time. The intended approach is:

- Use an LLM to work through the steps below. This will take real back-and-forth — budget time for it.
- Once you have a solution, make sure you actually understand it by verifying examples and working through the proof logic yourself.
- Write up the deliverables. LLM help is fine for code and for studying the math — but the writeup has to be yours, and you will need to defend it live.

LLM STEPS

Work through these in order. Each step is a starting point for a prompt — adapt the wording, try follow-ups, push back when the answer seems wrong.

LLM-0 Cold attempt

Give the LLM the problem statement as written. Read the answer carefully. **Does it actually solve the general case, or only specific instances?** *Test it on inputs you pick yourself. Many first answers to this problem are wrong or incomplete — your job here is to notice that before moving on.*

LLM-1 What does repeated pouring do?

Set the goal aside. Ask: **if I keep pouring into jar A over and over, what happens to its value?** *What is the total water drawn from the source after k pours? Is there a closed form? Work one small example by hand before trusting the LLM's answer.*

LLM-2 Subtracting a chosen amount from B

Now ask: **can I make jar B lose exactly qA total, for an integer q I choose?** *What if at each step I get to pick whether to pour from B or from C into A — does that freedom help? Try constructing sequences for q = 5, 7, 13, and a large q with a small C. Ask the LLM to find the general pattern and verify it on your examples.*

LLM-3 Choosing q and proving termination

You can now subtract any multiple of A from B. **Which multiple should you pick** *so the jars are in a strictly simpler state afterward? Does this remind you of any classical algorithm? Ask the LLM to state a loop invariant and prove: (i) each round produces legal pours, (ii) something strictly*

decreases, (iii) termination implies the goal is reached. Check the proof on your examples — LLM termination proofs on this problem are often hand-wavy.

DELIVERABLES

Q-1 Key ideas `write yourself`

A short informal explanation — half a page, no formal notation — of how the algorithm works. Imagine explaining it at a whiteboard to a sharp friend. Cover: why the goal simplifies, what repeated pouring gives you, how the third jar is used, and why the process terminates. This should be written entirely in your own words.

Q-2 Pseudocode and code `LLM ok for code` `pseudocode yourself`

Write the pseudocode yourself (not from LLM output). Then implement it in Python or Java — LLM help is fine here. Run on all four examples below and include the full pour sequence and final state for each.

Input (A, B, C)	Notes
`(3, 28, 100)`	Running example from class
`(7, 11, 23)`	Values close together
`(1, 1023, 1025)`	Large values; stress test
`(your choice, all ≥ 500)`	Pick something non-trivial

Q-3 Math writeup `write yourself`

Study whatever the LLM produces, but write every sentence yourself. Four sub-parts, each graded independently: