

Submitted to: Sir Rasikh Ali

Submitted by: Dua Saeed

Subject: PAI (Lab)

Section: BSAI-4A

Roll number: 042

**Water Jug with DFS & printing rules**

The Water Jug Problem involves two jugs with different capacities, and the goal is to measure a specific amount of water using these jugs. This implementation uses **Depth-First Search (DFS)** to find a valid sequence of operations.

**Step-by-Step Explanation:**

**1. Checking Validity of a State**

* A function is\_valid(state, capacities) ensures that the water level in each jug stays within its respective capacity.
* This prevents invalid states like overfilling or negative water levels.

**2. Generating Possible Successor States**

* The function get\_successors(state, capacities) generates all valid next states by performing these actions:
  + **Filling Jug X completely**
  + **Filling Jug Y completely**
  + **Emptying Jug X completely**
  + **Emptying Jug Y completely**
  + **Pouring water from X to Y until Y is full or X is empty**
  + **Pouring water from Y to X until X is full or Y is empty**
* Only valid states are added to the successor list.

**3. Depth-First Search (DFS) Implementation**

* The function dfs\_water\_jug(capacities, target) starts from **(0,0)**, representing empty jugs.
* A **stack** is used to explore paths deeply before backtracking.
* A **visited set** stores explored states to avoid infinite loops.
* The algorithm proceeds as follows:
  + **Pop a state from the stack** and mark it as visited.
  + **If the target amount is found in either jug, the solution path is printed.**
  + **Otherwise, generate successors and push them onto the stack.**
  + **If no solution is found, print "No solution found."**

**Why DFS?**

* **Explores deeper solutions first** before backtracking.
* **Less memory usage** compared to BFS since it does not store all nodes at once.
* **Efficient for smaller problems** like this water jug puzzle.

**Corrected Rules (5 & 6):**

* **Rule 5 (Pour from X to Y):** Transfer as much as possible from X to Y without exceeding Y’s capacity.
* **Rule 6 (Pour from Y to X):** Transfer as much as possible from Y to X without exceeding X’s capacity.
* 