**Lab 03(Water Jug Problems)**

****

**Name Syed M Haider Jafir**

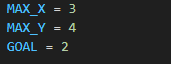
**Roll no SU92-BSAIM-S24-004**

**Section BSAI-4A**

**Subject PF(AI Lab)**

This Python program solves the classic **Water Jug Problem** using **Depth-First Search (DFS)**. The goal is to measure exactly 2 gallons using a 3-gallon jug and a 4-gallon jug. Let’s break it down step by step:

**Problem Setup**

****

MAX\_X and MAX\_Y define the capacities of the two jugs.

GOAL is the target amount of water we want in either jug.

**get\_successors(state) Function**

This function generates all possible next states from the current state (x, y) based on the allowed operations:

****

It checks and adds transitions for:

* Fill X: (MAX\_X, y)
* Fill Y: (x, MAX\_Y)
* Empty X: (0, y)
* Empty Y: (x, 0)
* Pour all X → Y (overflow allowed): pour all water from X into Y, even if Y overflows.
* Pour all Y → X (overflow allowed): same logic, reversed.
* Pour X → Y until Y is full: partial transfer.
* Pour Y → X until X is full: partial transfer.

**dfs(state, visited, path) Function**

This is the recursive **Depth-First Search** algorithm.

**Parameters:**

* state: current jug state (x, y)
* visited: set of visited states to avoid cycles
* path: list of steps taken so far

**Logic:**

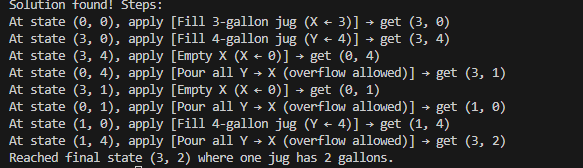
1. **Base case**: if either jug has GOAL gallons, return True.
2. **Mark current state as visited**.
3. **Explore all successors**:
   * Skip already visited states.
   * Add the transition to path.
   * Recursively call dfs on the new state.
   * If the recursive call fails, backtrack by removing the last step from path.

**solve(start=(0, 0)) Function**

This is the entry point of the program.

1. Initializes visited and path.
2. Calls dfs from the starting state (0, 0).
3. If a solution is found:
   * Prints each step with the rule applied.
   * Shows the final state where the goal is reached.

**Sample Output**

If successful, it prints something like:

**Summary**

This code:

* Models the water jug problem as a state space.
* Uses DFS to explore all possible paths.
* Stops when it finds a state where one jug has exactly 2 gallons.
* Prints the sequence of operations that lead to the solution.