

The Superior University

|  |  |  |
| --- | --- | --- |
| Name: Muhammad Bilal Sajid | Roll No: 018 | Course: PAI Lab |
| Semester: 4th | Section: BSAI 4A | Department: |
| Submitted To: | Total Marks: | Date: |

**Lab 3**

**Task: WaterJug with DFS & printing rules (also correct the rule 5 & 6)**

**Water Jug Problem using Depth-First Search (DFS)**

**Code Summary:**

The provided Python program solves the classic Water Jug Problem using Depth-First Search (DFS). The goal is to determine a sequence of actions that allow measuring an exact target amount of water using two jugs with given capacities.

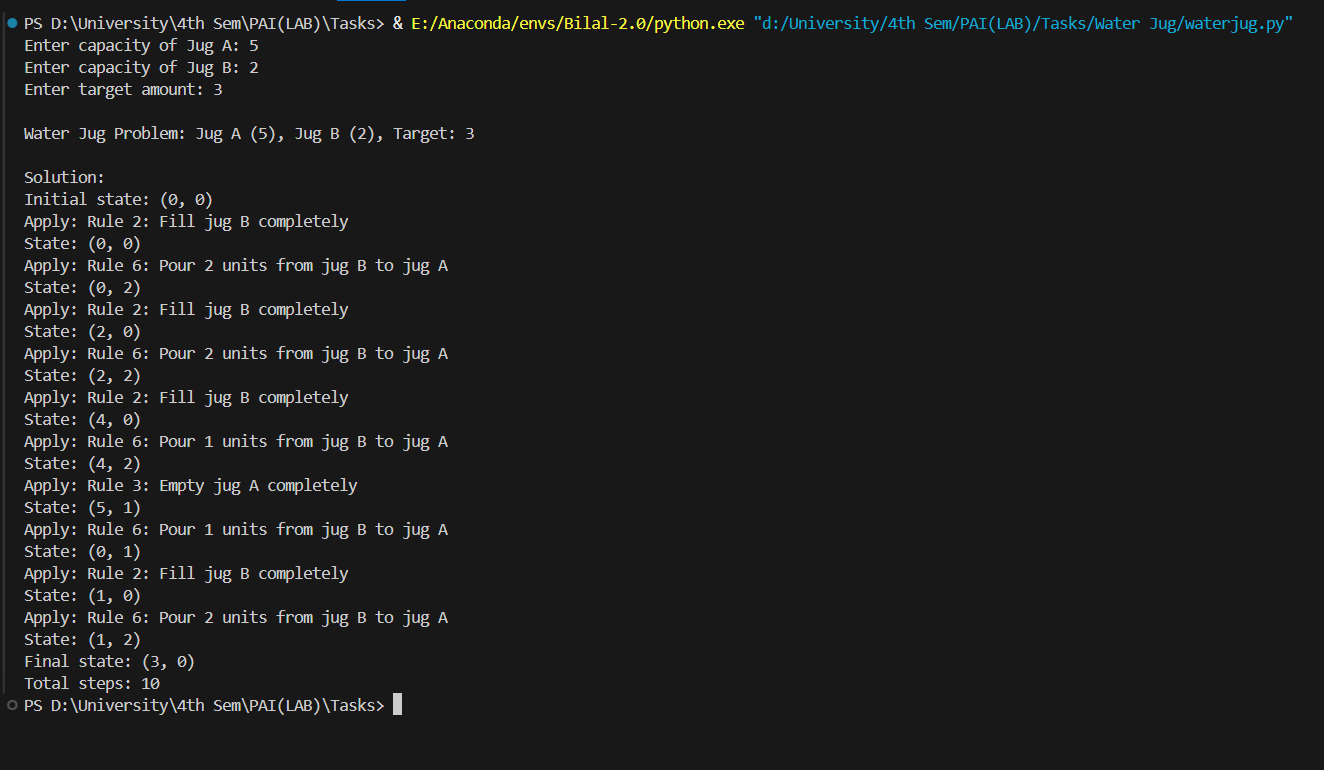
**How the Code Works:**

1. **State Representation:**
   * The current state of the jugs is represented as a tuple (A, B), where A and B denote the amount of water in Jug A and Jug B, respectively.
2. **Depth-First Search (DFS) Implementation:**
   * The function water\_jug\_dfs() initializes an empty stack with the starting state (0,0) and explores possible actions to reach the target amount.
   * A set visited keeps track of visited states to prevent cycles.
   * The algorithm pops the top element from the stack and checks if either jug contains the target amount.
   * If the maximum depth is reached, it stops further exploration.
   * Otherwise, the function get\_next\_states() generates all valid next states based on possible operations (filling, emptying, and pouring water between jugs).
   * The algorithm continues until a solution is found or all possibilities are exhausted.
3. **Generating Next States:**
   * get\_next\_states() defines six possible operations:
     1. Fill Jug A completely.
     2. Fill Jug B completely.
     3. Empty Jug A.
     4. Empty Jug B.
     5. Pour water from Jug A to Jug B.
     6. Pour water from Jug B to Jug A.
   * Each operation creates a new state and is added to the DFS stack if it has not been visited before.
4. **Printing the Solution:**
   * The function print\_solution() displays the step-by-step sequence of operations leading to the goal.
   * If no solution exists, it prints "No solution found!"
5. **User Input & Execution:**
   * The main() function takes user input for the capacities of both jugs and the target amount.
   * Calls water\_jug\_dfs() to find a solution and prints the steps.

**Why this Approach?**

* **DFS** is used to explore all possible solutions in a depth-wise manner. Although DFS may not always find the shortest path, it is effective for exploring possibilities in problems with a manageable state space.
* The algorithm ensures that all valid operations are considered while preventing redundant computations using a visited set.

**Example Output Screenshot:**

****