

**NAME: SABIKA ZAHRA**

**ROLL NUMBER: 074**

**CLASS: BSAI**

**SECTION: ALPHA**

**SEMESTER: 4**

**SUBMITTED TO: SIR RASIKH**

**DATE: 2,27,2025**

**Water Jug Problem Solution Report**

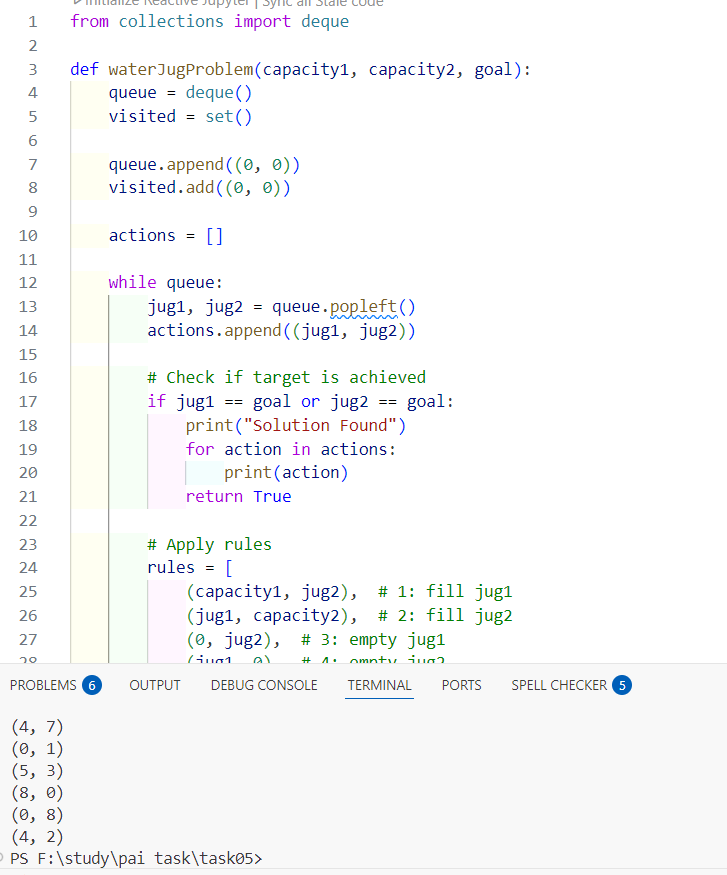
**Problem Statement**

The Water Jug Problem involves two jugs with fixed capacities and the goal of obtaining a specific amount of water using various permissible operations. In this implementation, we solve the problem using **Breadth-First Search (BFS)** to explore all possible states of the jugs efficiently.

**Approach Used**

The solution follows a BFS approach using a queue and a set to track visited states. The key steps include:

1. **Initialization:**
   * A queue (deque) is used to store jug states.
   * A set (visited) is used to avoid redundant computations.
   * The initial state (0,0) (both jugs empty) is added to the queue.
2. **State Exploration:**
   * While the queue is not empty, extract the front state (jug1, jug2).
   * If the target amount is found in either jug, print the solution steps and terminate.
   * Generate new states by applying the following rules:
     1. Fill jug1 to its maximum capacity.
     2. Fill jug2 to its maximum capacity.
     3. Empty jug1.
     4. Empty jug2.
     5. Pour water from jug1 to jug2 until jug2 is full or jug1 is empty.
     6. Pour water from jug2 to jug1 until jug1 is full or jug2 is empty.
     7. Pour all water from jug1 into jug2.
     8. Pour all water from jug2 into jug1.
   * If a new state hasn't been visited before, add it to the queue.
3. **Termination:**
   * If the queue is exhausted without finding the target, print "No Solution Found".

****

Since BFS explores all possible states before making a decision, the time complexity is **O(V + E)**, where V is the number of valid jug states and E is the number of transitions between states. This ensures an optimal solution if one exists.

**Conclusion**

The BFS-based approach ensures that we find the shortest path to reach the desired water quantity. By maintaining a visited set, redundant calculations are minimized, making the algorithm efficient. This implementation effectively demonstrates the application of state-space search in problem-solving.