

**The Superior University**

***Session 2023-2025***

***Department of Software Engineering***

***Faculty of Computer Science & Information Technology***

***The Superior University, Lahore***

***Course: Programming For Artificial Intelligence***

***Course Instructor:Sir Rasikh ali***

***Semester 4 BSAI***

***Fall 2025***

***Name:Sara hanif***

***Roll no:082***

***LAB TASK 3***

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

**Function Overview:**

The dfs\_water\_jug function determines whether it's possible to measure exactly goal liters of water using two jugs with capacities capacity1 and capacity2. It employs **Depth-First Search (DFS)** to explore all possible states.

### ****Key Components and Data Structures****

***Stack :***

* + Used to store different water levels in the jugs as states.
  + Ensures that DFS explores deeper paths first (LIFO behavior).

***Set:***

* + Keeps track of visited states to prevent redundant calculations.
  + Ensures efficient exploration by avoiding infinite loops.

***List :***

* + Records the sequence of actions leading to the solution.
  + Helps visualize the steps taken to reach the goal.

***Step-by-Step Execution***

def dfs\_water\_jug(capacity1, capacity2, goal):

    stack = []

    visited = set()

    stack.append((0, 0))

    visited.add((0, 0))

    actions = []

    while stack:

        jug1, jug2 = stack.pop()

        actions.append((jug1, jug2))

        if jug1 == goal or jug2 == goal:

            print("\n Solution Found using DFS:")

            for action in actions:

                print(action)

            return True

        possible\_moves = [

            (capacity1, jug2),

            (jug1, capacity2),

            (0, jug2),

            (jug1, 0),

            (jug1 - min(jug1, capacity2 - jug2), jug2 + min(jug1, capacity2 - jug2)),

            (jug1 + min(jug2, capacity1 - jug1), jug2 - min(jug2, capacity1 - jug1)),

        for state in possible\_moves:

            if state not in visited:

                visited.add(state)

                stack.append(state)

    print("\n❌ No Solution found using DFS")

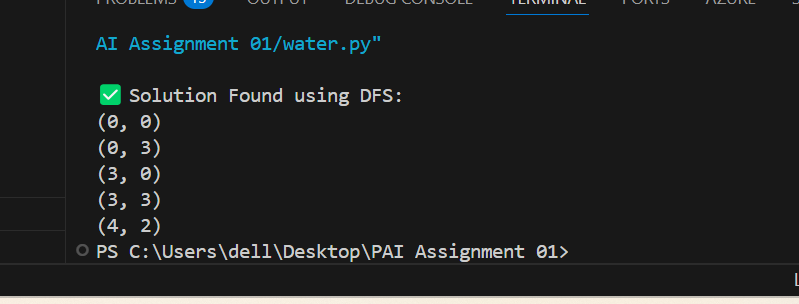
    return False

jug1Capacity = 4

jug2Capacity = 3

target = 2

dfs\_water\_jug(jug1Capacity, jug2Capacity, target)

******