

**Superior University Gold Campus**

**PAI Lab Task #2**

Name: Syed Ejiz Ul Hassan Kazmi

Roll. No: SU92-BSDSM-F23-025

Section: BSDS-4A

Instructor: Sir Rasikh

## **1. Problem Overview**

The **Water Jug Problem** is a classic AI problem where two jugs with fixed capacities must measure an exact amount of water using allowed operations. The problem is solved using **search algorithms**.

## **2. Techniques Used**

* **Breadth-First Search (BFS)**: Explores all possible states level by level.
* **Depth-First Search (DFS)**: Explores paths deeply before backtracking.
* **Queue (Deque)**: Used in BFS for level-order exploration.
* **Set (Visited States)**: Avoids re-exploring previously visited states.

## **3. BFS Implementation**

### ****3.1 BFS Algorithm****

1. **Start with empty jugs** (0,0).
2. **Use a queue (deque)** to store states and their rules.
3. **Apply all possible operations** (fill, empty, pour).
4. **Track visited states** to avoid loops.
5. **Stop if the goal amount is found** in any jug.

**3.3 Explanation of BFS Code**

* The queue stores **states** ((jug1, jug2)) and the **rule applied**.
* The loop removes a state, checks if the goal is reached, and applies **six valid rules**:
  + **Fill Jug 1**
  + **Fill Jug 2**
  + **Empty Jug 1**
  + **Empty Jug 2**
  + **Pour Jug 1 → Jug 2**
  + **Pour Jug 2 → Jug 1**
* If a new state has not been visited, it is added to the queue.

## **4. DFS Implementation**

### ****4.1 DFS Algorithm****

1. **Use a recursive function** to explore each path deeply.
2. **Maintain a set of visited states** to prevent infinite loops.
3. **Apply the same six rules** as BFS.
4. **Stop when the goal amount is reached**.

### ****4.3 Explanation of DFS Code****

* **Recursive function (dfs)** explores all possibilities until it finds a solution or backtracks.
* **Maintains visited states** to prevent loops.
* **Stores actions** taken at each step for tracing.
* **Applies the same six rules** as BFS.

1. **Comparing BFS and DFS:**

**BFS is preferred** when an optimal solution is required.  
 **DFS is faster** but may explore inefficient paths.

## **6. Conclusion**

The notebook implements **BFS and DFS** to solve the **Water Jug Problem**. The approach involves:

1. **Defining the problem** and rules.
2. **Applying BFS using a queue** for optimal solutions.
3. **Using DFS recursion** for deep exploration.
4. **Ensuring valid water transfer rules**.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_