

|  |  |
| --- | --- |
| **Student Name** | **Arun Adhikari** |
| **SRN No** | 202100406 |
| **Roll No** | 31 |
| **Program** | **Department of Computer Engineering** |
| **Year** | 2024-2025 |
| **Division** | G |
| **Subject** | AI LAB |
| **Assignment No** | 2 |

**Introduction**

The Water Jug Problem is a classic puzzle where the objective is to measure a specific amount of water using two jugs of different capacities. The puzzle is typically presented with two jugs and an unlimited supply of water, but the challenge is to measure exactly the required amount of water by filling, pouring, and emptying the jugs. This problem is a practical application of search algorithms in problem-solving.

In this project, we have used two well-known uninformed search algorithms, **Depth-First Search (DFS)** and **Breadth-First Search (BFS)**, to solve the Water Jug Problem and implemented a graphical user interface (GUI) to visualize the solution in real-time.

**Objective**

The goal of this project is to implement an efficient solution to the Water Jug Problem using DFS and BFS. The GUI allows users to input jug capacities and the target water amount and visualizes the solution steps with attractive colors representing the water level

**Problem Description**

You are given two jugs with different capacities, and an unlimited supply of water. Your task is to measure a specific amount of water, using only the two jugs and the following actions:

1. **Fill a jug**: Fill either jug completely.
2. **Empty a jug**: Empty either jug.
3. **Pour water from one jug to the other**: Pour as much water as possible from one jug into the other without overflowing.

The goal is to reach the target amount of water in either of the jugs.

**Search Algorithms**

To solve the problem, we employ two algorithms:

**Depth-First Search (DFS)**

DFS is an uninformed search strategy where the algorithm explores as far as possible along each branch before backtracking. In DFS, we start from the initial state (both jugs empty) and explore all possible states. The algorithm goes deep into a state (branch) until no further progress can be made, and then backtracks to explore the next state. The search continues until the target state is found or all possibilities are exhausted.

**Breadth-First Search (BFS)**

BFS, unlike DFS, explores all possible states at the present depth level before moving on to the next level. BFS starts from the initial state and explores all possible states level by level. It guarantees the shortest path to the goal because it explores all nodes at a distance d before exploring nodes at distance d+1.

**Explanation of Algorithms**

1. **DFS Algorithm**:
   * The DFS algorithm begins from the initial state (both jugs empty) and recursively explores all possible states until the goal is found.
   * We maintain a stack to store the current state, and at each step, we check all possible transitions from the current state (filling, emptying, pouring between jugs).
   * If we encounter a state that has already been visited, we backtrack to avoid cycles.
2. **BFS Algorithm**:
   * BFS begins from the initial state and explores all possible states in breadth-first order. This means that all states at a certain "depth" are explored before any deeper states are considered.
   * We maintain a queue to store states as they are discovered. Once the goal state is found, the algorithm terminates.
   * BFS guarantees that the first time it reaches the goal state, it will have found the shortest path.

**Algorithm Implementation**

The implementation consists of the following steps:

1. **GUI Setup**:
   * The GUI is built using the Tkinter library in Python. It allows users to input the jug capacities and the target amount of water.
   * Two jugs are represented as rectangles on a canvas. The water levels are visualized in blue.
2. **State Transitions**:
   * States are represented as tuples of the form (jug1, jug2), where jug1 and jug2 represent the current water levels in the two jugs.
   * From each state, we generate the next possible states by performing actions such as filling, emptying, or pouring between the two jugs.
3. **Search Algorithm Execution**:
   * Users can select either DFS or BFS to solve the problem.
   * The algorithm explores the possible states step by step, and each step is displayed on the GUI by updating the water levels in the jugs.
4. **Result Display**:
   * Once the solution is found, the algorithm displays the steps taken to reach the goal state on the GUI, along with the visualization of the water levels in the jugs at each step.

**Example: Jug 1 Capacity = 4, Jug 2 Capacity = 3, Target = 2**

Let’s consider a case where:

* **Jug 1 Capacity** = 4 liters
* **Jug 2 Capacity** = 3 liters
* **Goal** = 2 liters

The steps to solve this problem using the search algorithms would be as follows:

**Using DFS:**

* **Step 1**: Initial state (0, 0) → Both jugs are empty.
* **Step 2**: Fill Jug 1 → (4, 0)
* **Step 3**: Pour Jug 1 into Jug 2 → (1, 3) (Jug 1 gives 3 liters to Jug 2, and 1 liter remains in Jug 1).
* **Step 4**: Empty Jug 2 → (1, 0)
* **Step 5**: Pour Jug 1 into Jug 2 → (0, 1) (Jug 1 gives 1 liter to Jug 2).
* **Step 6**: Fill Jug 1 → (4, 1)
* **Step 7**: Pour Jug 1 into Jug 2 → (2, 3) (Jug 1 gives 2 liters to Jug 2, and 2 liters remain in Jug 1) → Goal is reached!

**Using BFS:**

* BFS explores all possible paths level by level and finds the shortest path to reach the goal. The states are explored step by step, with the first solution found being the optimal one.

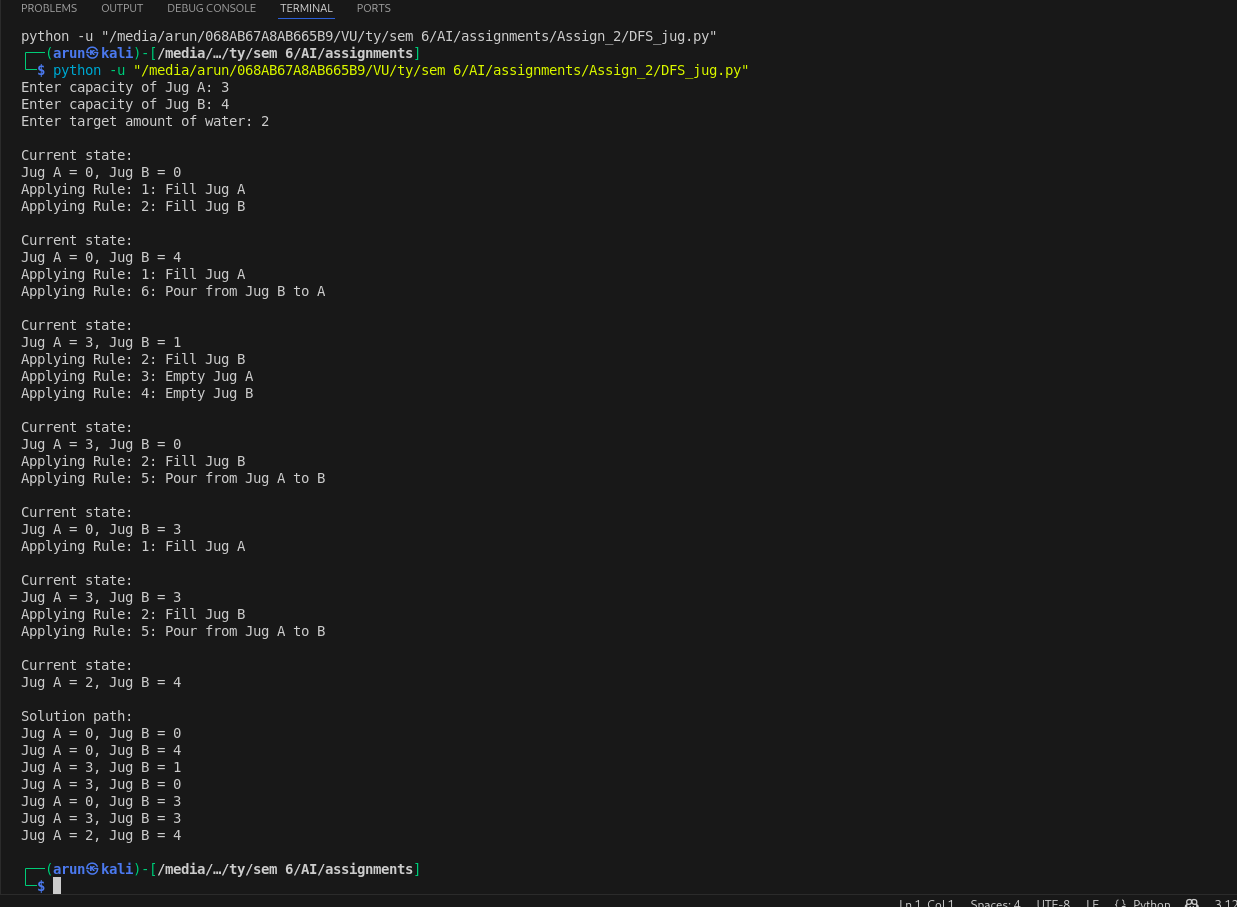
**GUI Visualization**

The graphical interface visualizes the solution steps as follows:

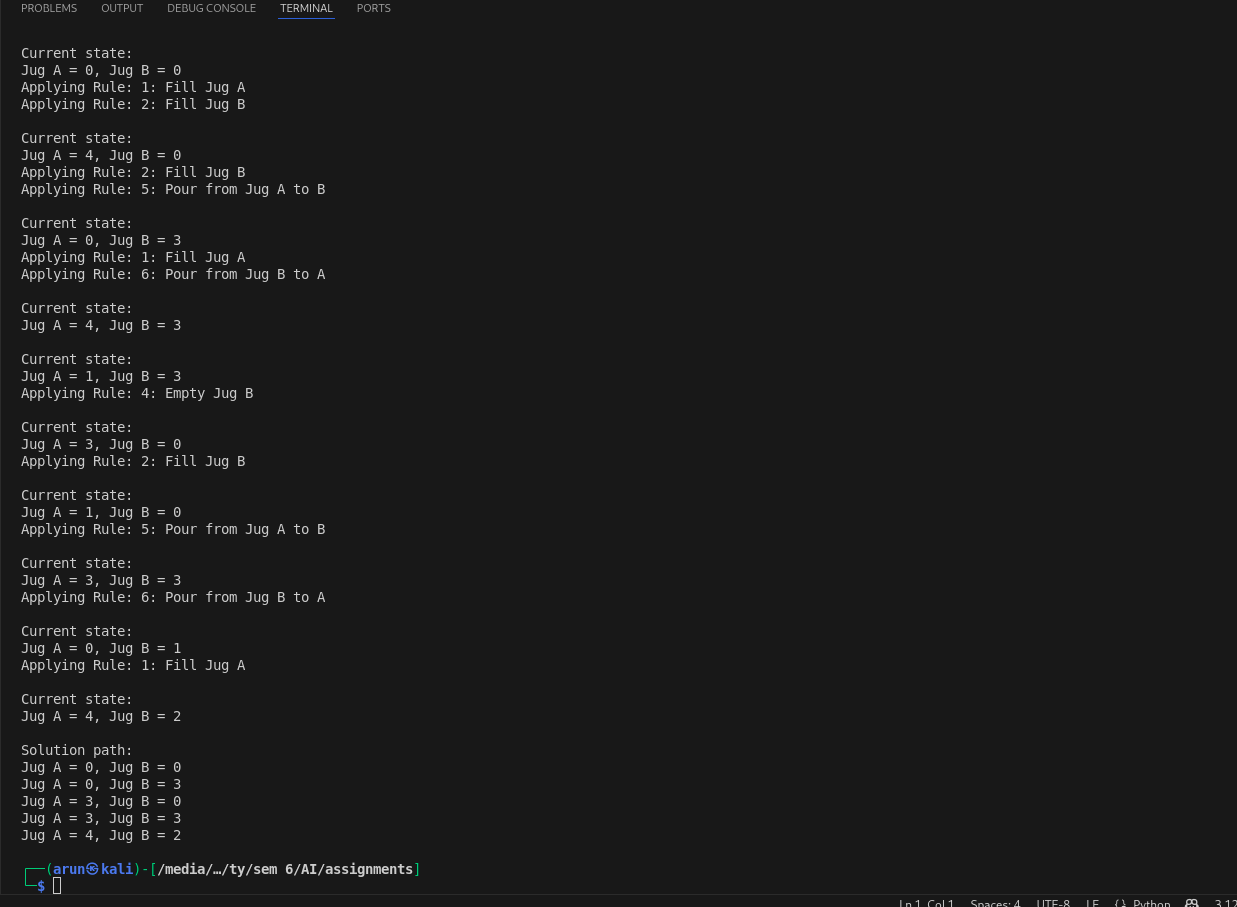
1. The two jugs are displayed as rectangles on the canvas.
2. The water levels are represented by filling the rectangles with blue color.
3. Each step in the solution is shown by updating the water levels in the jugs.
4. The algorithm continues updating the jugs' states until the goal is reached.

**Output :-**

* 1. **DFS**



**BFS**



**Conclusion**

In conclusion, the Water Jug Problem Solver successfully uses DFS and BFS algorithms to find solutions to the problem. The GUI implementation provides a visual representation of the water jug states at each step, making it easier to understand how the algorithms work. The DFS algorithm explores the solution space deeply, while BFS guarantees the shortest path to the goal.

This project demonstrates the power of search algorithms in problem-solving, and the GUI provides an interactive way for users to explore different scenarios.