

## 

***Water Jug Problem :***

*Result File*

**LAB 03: Water Jug Problem**

**Introduction**

In this project I am implementing the Water Jug Problem using Depth-First Search (DFS). The objective is to determine whether it is possible to measure a specific amount of water using two jugs with given capacities. The solution follows a state-space search approach with DFS.

**Code Explanation**

1. Depth-First Search (DFS) Implementation

The function water\_jug\_with\_dfs(capacity\_a, capacity\_b, target, max\_depth=1000) initializes the problem with an empty state (0,0).

A stack is used for DFS traversal, where each state is stored along with the path taken.

If the target amount is reached in either jug, the solution is returned.

If the search depth exceeds max\_depth, the algorithm continues without expanding further.

The function get\_next\_states(state, capacity\_a, capacity\_b) generates all possible next states based on predefined rules.

**2. State Transition Rules**

The function get\_next\_states() applies six possible operations:

Fill Jug A completely

Fill Jug B completely

Empty Jug A completely

Empty Jug B completely

Pour water from Jug A to Jug B (without overflowing)

Pour water from Jug B to Jug A (without overflowing)

**3. Solution Display**

The function print\_solution(solution) prints the sequence of steps taken to achieve the goal.

If no solution exists, it prints "No solution found."

The total number of steps required is displayed at the end.

**4. User Input and Execution**

The function main() takes user input for jug capacities and the target amount.

It calls water\_jug\_with\_dfs() to compute the solution and prints the results.

**Conclusion**

This project successfully implements DFS to solve the Water Jug Problem. The algorithm explores possible states systematically, ensuring a valid path to the target. The approach demonstrates how search algorithms can be applied to constraint-based problems.

The output is given below

