TASK 03

Implement Water Jug Problem using DFS

Code Explanation

Initialization:

- Two jugs of capacity cap1 and cap2 are given.
- The **goal** is to measure an exact amount of water.
- A **stack** is used for DFS, initialized with (0, 0).
- A **set** tracks visited states to avoid loops.
- A **dictionary (parents)** stores previous states for backtracking.

Rules:

The algorithm considers the following actions at each step:

- Fill jug1 completely (to capacity).
- Fill jug2 completely (to capacity).
- Empty jug1 completely**.**
- Empty jug2 completely.
- Pour water from jug1 to jug2 until jug2 is full or jug1 is empty.
- Pour water from jug2 to jug1 until jug1 is full or jug2 is empty.
- Transfer all water from jug1 to jug2 (if possible).
- Transfer all water from jug2 to jug1 (if possible).

DFS Execution:

- The algorithm follows a **LIFO** approach using a stack.
- It **removes (pops)** the last state added and checks if the goal is reached.
- If the goal is reached, it traces back the solution using the parent dictionary.
- If a new state has not been visited, it is:
    - Added to the stack for future exploration.
    - Marked as visited to prevent revisiting.
    - Linked to its parent for backtracking later.

Goal Check:

If a state contains the desired amount of water:

- The algorithm retraces the steps from the goal state to (0,0).
- The sequence of steps is stored in **actions** and reversed to get the correct order.
- The list of steps is returned as the solution.

If No Solution Exists:

- If the stack is empty and no valid solution is found, the function returns None.

Output: