Water Jug

Aim

To implement DFS algorithm to water jug

Algorithm

1. Start
2. Initiate starting state with both jugs empty
3. Mark current state as visited.
4. If amount of water is equal in both jugs print the solution and terminate
5. Generate all possible next states
6. Recursively apply DFS to each unvisited state
7. Stop

```
def waterJugDFS(jug1_capacity, jug2_capacity, target,
                current_state = None, visited = None):
    if visited is None:
        visited = set()
    if current_state is None:
        current_state = (0,0)
    if current_state in visited:
        return False;
    visited.add(current_state)
    jug1, jug2 = current_state
    print(f"Jug1: {jug1}, Jug2: {jug2}")
    if jug1 == target or jug2 == target:
        print("Solution found!")
        return True
    possible_moves = [
        (jug1_capacity, jug2),
        (jug1, jug2_capacity),
        (0, jug2),
        (jug1, 0),
        (min(jug1_capacity, jug1 + jug2), max(0, jug2 - (jug1_capacity
                                                  - jug1))),
        (max(0, jug1 - (jug2_capacity - jug2)), min(jug2_capacity,
                                                  jug1 + jug2))
    ]
    for next_state in possible_moves:
        if waterJugDFS(jug1_capacity, jug2_capacity,
                       target, next_state, visited):
            return True
```

```
if __name__ == "__main__":
    jug1_capacity = 4
    jug2_capacity = 3
    target = 2
    print("DFS Traversal : ")
    if not waterJugDFS(jug1_capacity, jug2_capacity,
                       target):
        print("No solution found")
```

Output:

DFS Traversal

Jug1 : 0	Jug2 : 0
Jug1 : 4	Jug2 : 0
Jug1 : 4	Jug2 : 3
Jug1 : 0	Jug2 : 3
Jug1 : 3	Jug2 : 3
Jug1 : 4	Jug2 : 2

Result:

The program has been executed successfully and the output has been verified