Exp No : 03

Date : 24/08/24

Water Jug using DFS

Aim:

To solve Water Jug problem using DFS to determine a specific amount of water to be measured.

Algorithm:

1) Create a stack to store the state of the jugs.
2) Initialise the stack with initial state (both jugs empty).
3) While the stack is not empty, do the following
   i) Pop a state from the stack
   ii) If the state represents the desired quantity, stop and return the solution.
   iii) Generate all possible next states from the current state.
   iv) Push the next state onto the stack.
4) If the stack becomes empty and no solution is found then, the problem is unsolvable.

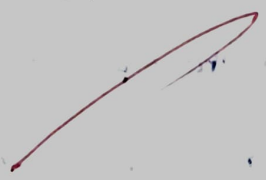
Program Code:

```
def SolveWaterJugProblem(capacity_jug1,
        capacity_jug2, desired_quantity):
    stack = []
    stack.append((0,0))
    while stack:
        current_state = stack.pop()
        if current_state[0] == desired_quantity
            or current_state[1] == desired_
                quantity
            return current_state
        next_states = generateNextStates(
                current_state,
            capacity_jug1,
                capacity_jug2):
            stack.extend(next_states)
    return "No solution found"
def generateNextStates(state, capacity_jug1,
                capacity_jug2):
    next_states=[]
    next_states.append((capacity_jug1,
                state[1]))
    next_states.append((state[0], capacity_jug2))
    next_states.append((0, state[1]))
    next_states.append((state[0], 0))
    pour_amount = min(state[0], capacity_jugs
                - state[1])
```

```
next_states.append((state[0] - pour_amount,
                    state[1] + pour_amount))

pour_amount = min(state[1], capacity_jug1 - state[0])

next_states.append((state[0] + pour_amount,
                    state[1] - pour_amount))

return next_states

solution = solve_WaterJugProblem(4,3,2)

print("Solution : ", solution)
```

Output :

```
Solution : (4,2)
```



Results:

Thus, the program of water jug using dfs is successfully executed and the output is verified.