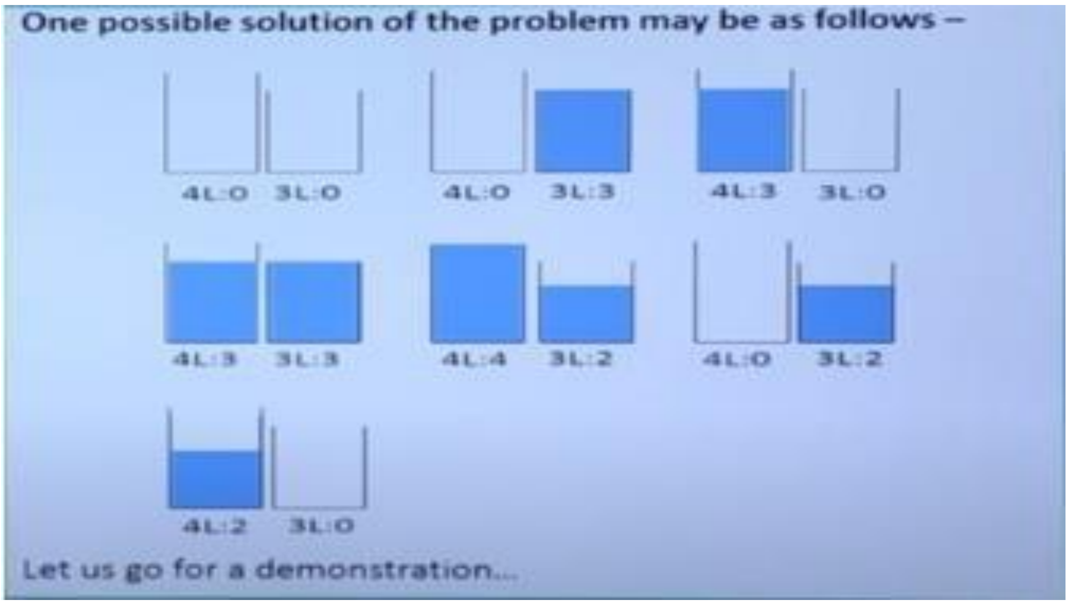
REG NO.220701055
EX.NO: 5 **DATE: 13.09.24**

DEPTH-FIRST SEARCH – WATER JUG PROBLEM

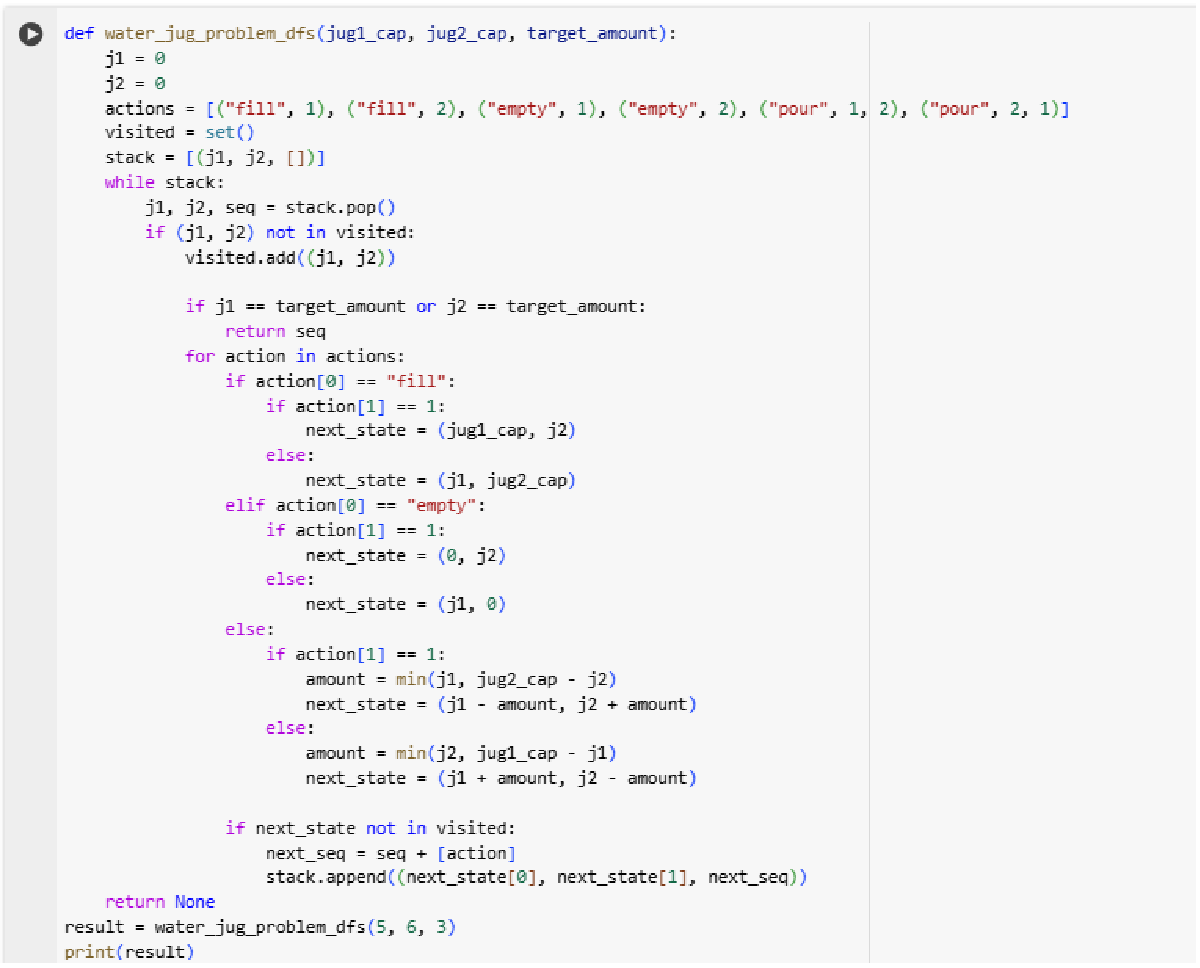
In the **water jug problem in Artificial Intelligence**, we are provided with two jugs: one having the capacity to hold 3 gallons of water and the other has the capacity to hold 4 gallons of water. There is no other measuring equipment available and the jugs also do not have any kind of marking on them. So, the agent's task here is to fill the 4-gallon jug with 2 gallons of water by using only these two jugs and no other material. Initially, both our jugs are empty.



One possible solution of the problem may be as follows –

Let us go for a demonstration...

CODE:

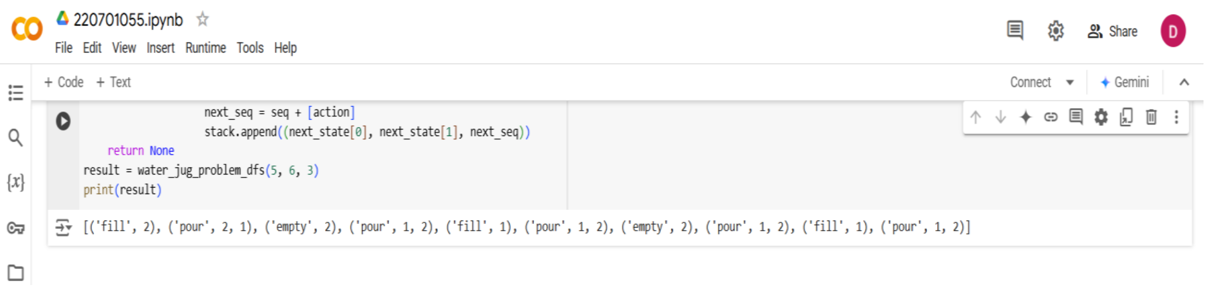


```python
def water_jug_problem_dfs(jug1_cap, jug2_cap, target_amount):
    j1 = 0
    j2 = 0
    actions = [("fill", 1), ("fill", 2), ("empty", 1), ("empty", 2), ("pour", 1, 2), ("pour", 2, 1)]
    visited = set()
    stack = [(j1, j2, [])]
    while stack:
        j1, j2, seq = stack.pop()
        if (j1, j2) not in visited:
            visited.add((j1, j2))

            if j1 == target_amount or j2 == target_amount:
                return seq
            for action in actions:
                if action[0] == "fill":
                    if action[1] == 1:
                        next_state = (jug1_cap, j2)
                    else:
                        next_state = (j1, jug2_cap)
                elif action[0] == "empty":
                    if action[1] == 1:
                        next_state = (0, j2)
                    else:
                        next_state = (j1, 0)
                else:
                    if action[1] == 1:
                        amount = min(j1, jug2_cap - j2)
                        next_state = (j1 - amount, j2 + amount)
                    else:
                        amount = min(j2, jug1_cap - j1)
                        next_state = (j1 + amount, j2 - amount)

                if next_state not in visited:
                    next_seq = seq + [action]
                    stack.append((next_state[0], next_state[1], next_seq))
    return None
result = water_jug_problem_dfs(5, 6, 3)
print(result)
```

OUTPUT:



```python
                    next_seq = seq + [action]
                    stack.append((next_state[0], next_state[1], next_seq))
    return None
result = water_jug_problem_dfs(5, 6, 3)
print(result)
```

```
[('fill', 2), ('pour', 2, 1), ('empty', 2), ('pour', 1, 2), ('fill', 1), ('pour', 1, 2), ('empty', 2), ('pour', 1, 2), ('fill', 1), ('pour', 1, 2)]
```

RESULT: thus the water jug DFS problem using python is executed successfully.