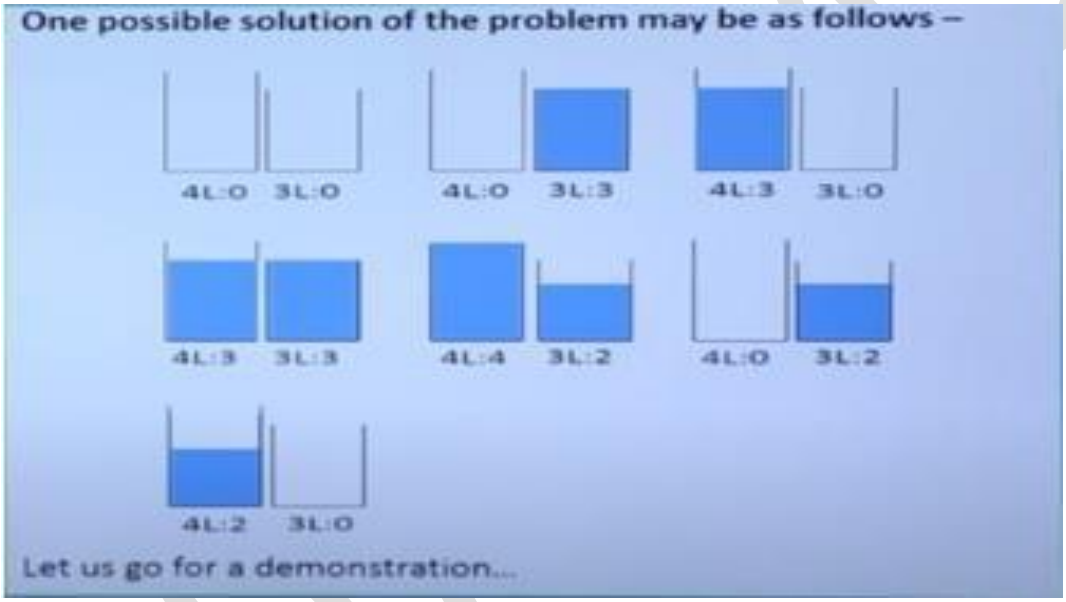
EX.NO: 3 **DATE: 23 - 08 - 2024**

DEPTH FIRST SEARCH-WATER JUG PROBLEM

AIM:

To implement **Water – jug problem** using depth first search algorithm.

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.



PROGRAM:

```
def fill_4_gallon(x, y, x_max, y_max):
    return (x_max, y)
def fill_3_gallon(x, y, x_max, y_max):
    return (x, y_max)
def empty_4_gallon(x, y, x_max, y_max):
    return (0, y)
def empty_3_gallon(x, y, x_max, y_max):
    return (x, 0)
def pour_4_to_3(x, y, x_max, y_max):
```

```
    transfer = min(x, y_max - y)
    return (x - transfer, y + transfer)
def pour_3_to_4(x, y, x_max, y_max):
    transfer = min(y, x_max - x)
    return (x + transfer, y - transfer)
def dfs_water_jug(x_max, y_max, goal_x, visited=None, start=(0, 0)):
    if visited is None:
        visited = set()
    stack = [start]
    while stack:
        state = stack.pop()
        x, y = state
        if state in visited:
            continue
        visited.add(state)
        print(f"Visiting state: {state}")
        if x == goal_x:
            print(f"Goal reached: {state}")
            return state
        next_states = [
            fill_4_gallon(x, y, x_max, y_max),
            fill_3_gallon(x, y, x_max, y_max),
            empty_4_gallon(x, y, x_max, y_max),
            empty_3_gallon(x, y, x_max, y_max),
            pour_4_to_3(x, y, x_max, y_max),
            pour_3_to_4(x, y, x_max, y_max)
        ]
        for new_state in next_states:
            if new_state not in visited:
                stack.append(new_state)
    return None
x_max = 4
y_max = 3
goal_x = 2
dfs_water_jug(x_max, y_max, goal_x)
```

OUTPUT:

220701043 - POAI - LAB EXPERIMENTS

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
dfs_water_jug(x_max, y_max, goal_x)
```

```
Visiting state: (0, 0)
Visiting state: (0, 3)
Visiting state: (3, 0)
Visiting state: (3, 3)
Visiting state: (4, 2)
Visiting state: (4, 0)
Visiting state: (1, 3)
Visiting state: (1, 0)
Visiting state: (0, 1)
Visiting state: (4, 1)
Visiting state: (2, 3)
Goal reached: (2, 3)
(2, 3)
```

RESULT:

Thus the water-jug problem is implemented successfully using depth-first search algorithm.