

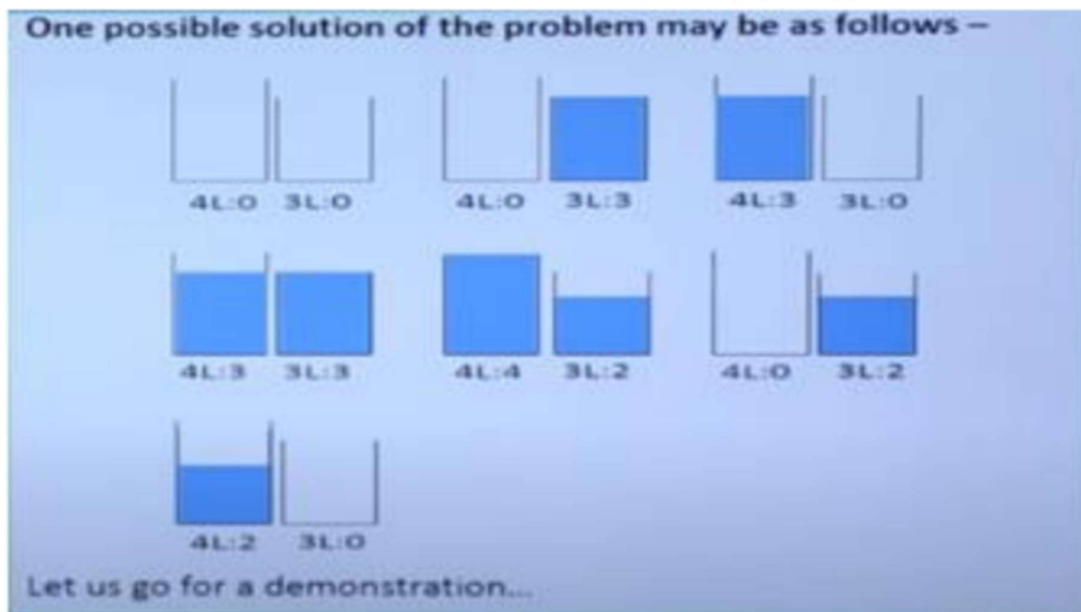
EX.NO:3

DATE:

REG NO:220701024

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.



CODE:

```
def water_jug_problem(jug1_capacity, jug2_capacity, target_capacity):
    """
    Solves the water jug problem using a breadth-first search algorithm.

    Args:
        jug1_capacity: The capacity of the first jug.
        jug2_capacity: The capacity of the second jug.
        target_capacity: The target amount of water to achieve in one of the jugs.

    Returns:
        A list of tuples representing the steps to achieve the target amount of water,
        or None if the target is not achievable.
    """

    visited = set()
    queue = [(0, 0), []] # [(jug1_amount, jug2_amount), [path]]

    while queue:
        (jug1_amount, jug2_amount), path = queue.pop(0)

        if jug1_amount == target_capacity or jug2_amount == target_capacity:
            return path + [(jug1_amount, jug2_amount)]

        if (jug1_amount, jug2_amount) in visited:
            continue
        visited.add((jug1_amount, jug2_amount))

        # Possible actions:
        # 1. Fill jug1
        queue.append([(jug1_capacity, jug2_amount), path + [(jug1_amount, jug2_amount)]])

        queue.append([(jug1_amount, jug2_capacity), path + [(jug1_amount, jug2_amount)]])
        # 3. Empty jug1
        queue.append([(0, jug2_amount), path + [(jug1_amount, jug2_amount)]])
        # 4. Empty jug2
        queue.append([(jug1_amount, 0), path + [(jug1_amount, jug2_amount)]])
        # 5. Pour water from jug1 to jug2
        pour_amount = min(jug1_amount, jug2_capacity - jug2_amount)
        queue.append([(jug1_amount - pour_amount, jug2_amount + pour_amount), path + [(jug1_amount, jug2_amount)]])
        # 6. Pour water from jug2 to jug1
        pour_amount = min(jug2_amount, jug1_capacity - jug1_amount)
        queue.append([(jug1_amount + pour_amount, jug2_amount - pour_amount), path + [(jug1_amount, jug2_amount)]])

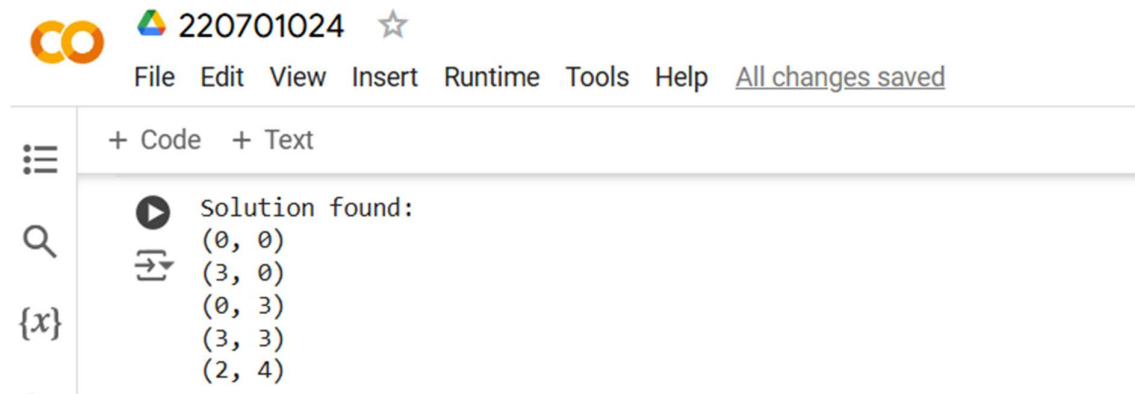
    return None

# Example usage:
jug1_capacity = 3
jug2_capacity = 4
target_capacity = 2

solution = water_jug_problem(jug1_capacity, jug2_capacity, target_capacity)

if solution:
    print("Solution found:")
    for step in solution:
        print(step)
else:
    print("No solution found.")
```

OUTPUT:



The screenshot shows a code editor with a top bar containing the 'CO' logo, the username '220701024', and a star icon. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a link 'All changes saved'. The editor area has a left sidebar with icons for a menu, search, and a variable '{x}'. The main area displays the output: 'Solution found:' followed by a list of coordinate pairs: (0, 0), (3, 0), (0, 3), (3, 3), and (2, 4). A large, light gray watermark '220701024' is diagonally across the lower half of the image.

```
CO 220701024 ☆
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Solution found:
(0, 0)
(3, 0)
(0, 3)
(3, 3)
(2, 4)
```

RESULT:

Thus the DEPTH-FIRST SEARCH – WATER JUG PROBLEM has been executed successfully.