**Task: Water Jug Problem**

**Objectives:**

Implement a Python program to solve the water jug problem by finding the sequence of steps to fill a 4-gallon jug with exactly 2 gallons of water

**Task Name:**

Given a 4-gallon jug (X) and a 3-gallon jug (Y), fill X with exactly 2 gallons of water

(assume an infinite amount of water is available)

• Implement the above-mentioned problem of water jug problem using Python.

\*You can implement using these rules or you can give It a try on your own

customized method.

**Rule 1: If X = 0 then X = 4 (fill X)**

**Rule 2: If Y = 0 then Y = 3 (fill Y)**

**Rule 3: If X > 0 then X = 0 (empty X)**

**Rule 4: If Y > 0 then Y = 0 (empty Y)**

**Rule 5: If X + Y >= 3 and X > 0 then X = X – (3 – Y) and Y = 3 (fill Y from X)**

**Rule 6: If X + Y >= 4 and Y > 0 then X = 4 and Y = Y – (4 – X) (fill X from Y)**

**Rule 7: If X + Y <= 3 and X > 0 then X = 0 and Y = X + Y (empty X into Y)**

**Rule 8: If X + Y <= 4 and Y > 0 then X = X + Y and Y = 0 (empty Y into X)**

**Code:**

def apply\_rule(state, rule):

    X, Y = state

    if rule == 1:  # *Rule 1: If X = 0 then X = 4 (fill X)*

        if X == 0:

            return (4, Y)

    elif rule == 2:  # *Rule 2: If Y = 0 then Y = 3 (fill Y)*

        if Y == 0:

            return (X, 3)

    elif rule == 3:  # *Rule 3: If X > 0 then X = 0 (empty X)*

        if X > 0:

            return (0, Y)

    elif rule == 4:  # *Rule 4: If Y > 0 then Y = 0 (empty Y)*

        if Y > 0:

            return (X, 0)

    elif rule == 5:  # *Rule 5: If X + Y >= 3 and X > 0 then X = X – (3 – Y) and Y = 3 (fill Y from X)*

        if X + Y >= 3 and X > 0:

            return (X - (3 - Y), 3)

    elif rule == 6:  # *Rule 6: If X + Y >= 4 and Y > 0 then X = 4 and Y = Y – (4 – X) (fill X from Y)*

        if X + Y >= 4 and Y > 0:

            return (4, Y - (4 - X))

    elif rule == 7:  # *Rule 7: If X + Y <= 3 and X > 0 then X = 0 and Y = X + Y (empty X into Y)*

        if X + Y <= 3 and X > 0:

            return (0, X + Y)

    elif rule == 8:  # *Rule 8: If X + Y <= 4 and Y > 0 then X = X + Y and Y = 0 (empty Y into X)*

        if X + Y <= 4 and Y > 0:

            return (X + Y, 0)

    return None

def solve\_water\_jug():

    # *Start with both jugs empty*

    initial\_state = (0, 0)

    # *List of rules to apply*

    rules = [1, 2, 3, 4, 5, 6, 7, 8]

    # *Keep track of the states we visit*

    visited\_states = set()

    # *Queue to keep track of states to explore*

    queue = [initial\_state]

    # *To keep track of the path for each state*

    parent = {initial\_state: None}

    while queue:

        state = queue.pop(0)  # *Get the first state from the queue*

        if state[0] == 2:  # *If the first jug has exactly 2 gallons, we found a solution*

            # *Backtrack to find the path*

            path = []

            while state is not None:

                path.append(state)

                state = parent[state]

            return path[::-1]  # *Return reversed path*

        # *If we have already visited this state, skip it*

        if state in visited\_states:

            continue

        visited\_states.add(state)  # *Mark this state as visited*

        # *Try applying each rule to the current state*

        for rule in rules:

            new\_state = apply\_rule(state, rule)

            if new\_state and new\_state not in visited\_states:

                queue.append(new\_state)

                parent[new\_state] = state  # *Record the parent state*

    return []  # *Return empty if no solution is found*

# *Run the solver and print the solution path*

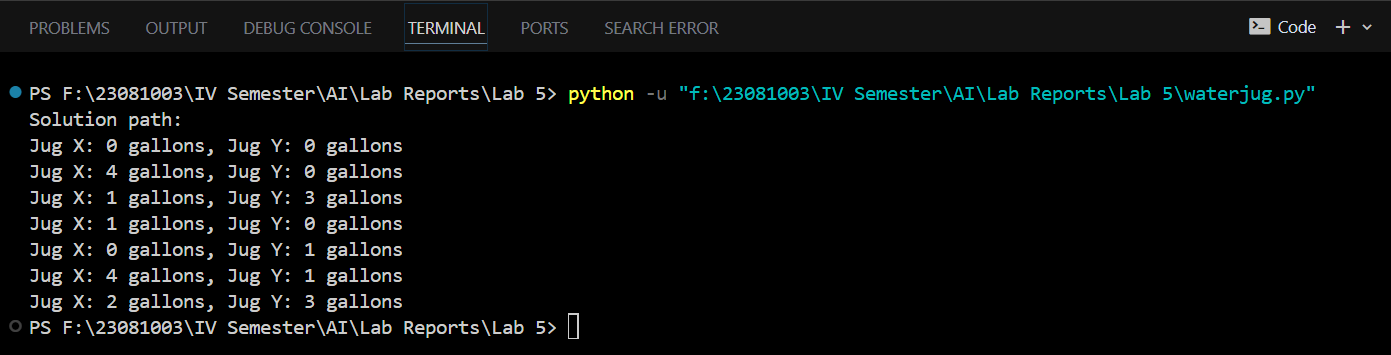
solution\_path = solve\_water\_jug()

print("Solution path:")

for state in solution\_path:

    print(f"Jug X: {state[0]} gallons, Jug Y: {state[1]} gallons")

**Output:**

****

**Conclusion:**

The program effectively finds the sequence of steps to get exactly 2 gallons of water in the 4-gallon jug (X) using systematic application of predefined rules. The Breadth-First Search ensures that the shortest path to the solution is found. By following the steps in the printed solution path, one can understand how to reach the desired goal starting from empty jugs.