**Experiment 3:**

**Aim:** To write a python program to implement a water jug program.

**Algorithm:**

1. Represent jugs' water levels as (x, y) with inputs for capacities, target, and initial state (0, 0).
2. Define moves: fill either jug, empty either jug, or pour water between jugs.
3. Use BFS to explore possible states while tracking visited states.
4. Stop when the target amount is reached and record the solution path.
5. Trace back and display the sequence of moves from the goal to the initial state

**Program:**

from collections import deque

# A utility function to check if the current state is valid

def is\_valid(x, y, max\_x, max\_y):

return 0 <= x <= max\_x and 0 <= y <= max\_y

# Function to find the sequence of steps to measure z liters

def water\_jug\_problem(max\_x, max\_y, z):

# Initial state (0, 0)

initial\_state = (0, 0)

# Queue for BFS

queue = deque([(initial\_state, [])]) # queue stores tuples of (state, operations)

# Set of visited states

visited = set()

visited.add(initial\_state)

# Perform BFS to find the solution

while queue:

(x, y), path = queue.popleft()

# If the current state is the goal state, return the path

if x == z or y == z:

print(f"Solution found! The steps are:")

for step in path:

print(step)

return

# List of possible state transitions (actions)

possible\_actions = [

(max\_x, y), # Fill the first jug

(x, max\_y), # Fill the second jug

(0, y), # Empty the first jug

(x, 0), # Empty the second jug

(x - min(x, max\_y - y), y + min(x, max\_y - y)), # Pour from jug 1 to jug 2

(x + min(y, max\_x - x), y - min(y, max\_x - x)), # Pour from jug 2 to jug 1

]

# Explore all possible state transitions

for next\_x, next\_y in possible\_actions:

if is\_valid(next\_x, next\_y, max\_x, max\_y) and (next\_x, next\_y) not in visited:

visited.add((next\_x, next\_y))

new\_path = path + [f"({next\_x}, {next\_y})"]

queue.append(((next\_x, next\_y), new\_path))

print(f"No solution found for measuring {z} liters.")

if \_\_name\_\_ == "\_\_main\_\_":

# Input from the user

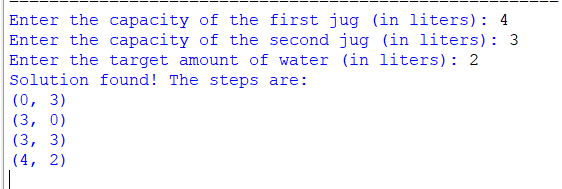
max\_x = int(input("Enter the capacity of the first jug (in liters): "))

max\_y = int(input("Enter the capacity of the second jug (in liters): "))

z = int(input("Enter the target amount of water (in liters): "))

water\_jug\_problem(max\_x, max\_y, z)

**OUTPUT:**

****

**Result:** Thus, the program was successfully completed using python programming.