

# **PRINCIPLES OF ARTIFICIAL INTELLIGENCE**

## **LABORATORY PROGRAMS**

### **WATER JUG PROBLEM:**

#### **SOURCE CODE:**

```
from collections import deque

def waterJugProblem(a, b, target):
    # Dictionary to track visited states
    visited = {}

    # Queue for BFS traversal
    queue = deque([(0, 0)])

    # Path to store the sequence of states
    path = []

    # Flag to indicate if the target is reachable
    isSolvable = False

    while queue:
        # Current state
        state = queue.popleft()

        # If state is already visited, skip
        if state in visited:
            continue

        # Mark state as visited
        visited[state] = True

        # Add state to path
        path.append(state)

        # Check if target is reached
```

```

    if state[0] == target or state[1] == target:
        isSolvable = True
        break

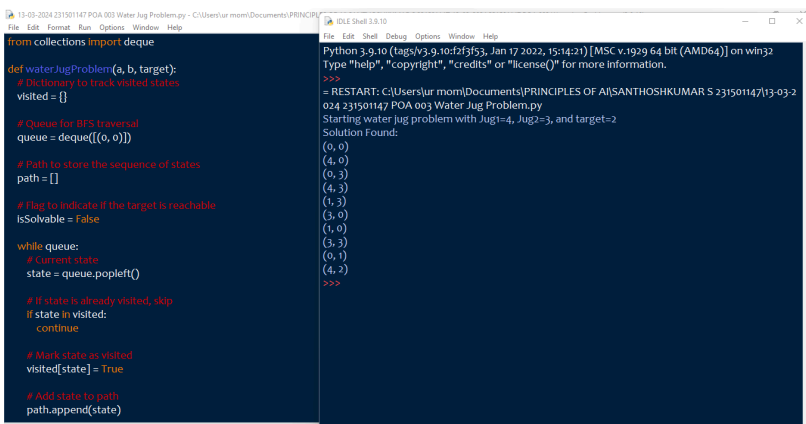
# Perform possible operations
for newState in [(a, state[1]), (state[0], b), (0, state[1]), (state[0], 0),
                 (min(state[0] + state[1], a), 0 if state[0] + state[1] <= a else state[1] - (a -
state[0])),
                 (0 if state[0] + state[1] <= b else state[0] - (b - state[1]), min(state[0] +
state[1], b))]:
    if newState not in visited:
        queue.append(newState)

# If solution is found, print path
if isSolvable:
    print("Solution Found:")
    for state in path:
        print(state)
else:
    print("No solution found.")

# Example usage
if __name__ == "__main__":
    Jug1, Jug2, target = 4, 3, 2
    print("Starting water jug problem with Jug1={}, Jug2={}, and target={}".format(Jug1,
Jug2, target))
    waterJugProblem(Jug1, Jug2, target)

```

## OUTPUT:



The screenshot shows a Python IDE with two windows. The left window displays the source code for the water jug problem, and the right window shows the output of the program.

**Source Code (Left Window):**

```

from collections import deque

def waterJugProblem(a, b, target):
    # Dictionary to track visited states
    visited = {}

    # Queue for BFS traversal
    queue = deque([(0, 0)])

    # Path to store the sequence of states
    path = []

    # Flag to indicate if the target is reachable
    isSolvable = False

    while queue:
        # Current state
        state = queue.popleft()

        # If state is already visited, skip
        if state in visited:
            continue

        # Mark state as visited
        visited[state] = True

        # Add state to path
        path.append(state)

```

**Output (Right Window):**

```

Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\ur mom\Documents\PRINCIPLES OF AI\SANTHOSH KUMAR S 231501147\13-03-2
024 231501147 POA 003 Water Jug Problem.py
Starting water jug problem with Jug1=4, Jug2=3, and target=2
Solution Found:
(0, 0)
(4, 0)
(0, 3)
(4, 3)
(4, 2)
(3, 0)
(1, 0)
(3, 3)
(0, 1)
(4, 2)
>>>

```