S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR

Practical No. 01

Aim: Apply the concept of production rules to solve the water jug problem. (A Water Jug Problem: You are given two jugs, a 4-gallon one and a 3-gallon one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the 4-gallon jug?)

Name of Student: Shruika Pradeep Bagdi

Roll No.: CS22130

Semester/Year: V/III

Academic Session: 2024-2025

Date of Performance:

Date of Submission:

AIM: Apply the concept of production rules to solve the water jug problem. (A Water Jug Problem: You are given two jugs, a 4-gallon one and a 3-gallon one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the 4-gallon jug?)

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- To be able to understand the concept of Water Jug.
- To be able to acquire the puzzle solving capability.

THEORY:
The water jug problem is a classic puzzle in artificial intelligence and computer science that involves solving a problem related to filling and measuring water using two or more jugs of known capacities. This problem serves as an interesting and practical example of how AI algorithms can be applied to find solutions for real-world problems.
The basic setup of the water jug problem typically involves two or more jugs of different capacities, with the goal of measuring or obtaining a specific amount of water. Here's a common version of the problem using two jugs:
Problem Statement:
You are given two jugs with capacities A liters and B liters, where A and B are positive integers. The jugs have no markings to indicate their current level of water. You also have access to a water source and a sink. Your task is to measure exactly C liters of water, where C is a positive integer, using these jugs.

- You can perform the following operations:
- Fill a jug from the water source.
- Empty a jug into the sink.

Pour the water from one jug into the other until either the source jug is empty or the target jug is full.
The objective is to find a series of operations that will result in having C liters of water in one of the jugs or determining that it is impossible to measure C liters with the given jugs and operations. 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.
So, to solve this problem, following set of rules were proposed:

Production rules for solving the water jug problem

Here, let ***x*** denote the 4-gallon jug and ***y*** denote the 3-gallon jug.

S.No.	**Initial State**	**Condition**	**Final state**	**Description of action taken**

1.	(x,y)	If x<4	(4,y)	Fill the 4 gallon jug completely
2.	(x,y)	if y<3	(x,3)	Fill the 3 gallon jug completely
3.	(x,y)	If x>0	(x-d,y)	Pour some part from the 4 gallon jug
4.	(x,y)	If y>0	(x,y-d)	Pour some part from the 3 gallon jug
5.	(x,y)	If x>0	(0,y)	Empty the 4 gallon jug
6.	(x,y)	If y>0	(x,0)	Empty the 3 gallon jug
7.	(x,y)	If (x+y)<7	(4, y-[4-x])	Pour some water from the 3 gallon jug to fill the four gallon jug
8.	(x,y)	If (x+y)<7	(x-[3-y],y)	Pour some water from the 4 gallon jug to fill the 3 gallon jug.
9.	(x,y)	If (x+y)<4	(x+y,0)	Pour all water from 3 gallon jug to the 4 gallon jug
10.	(x,y)	if (x+y)<3	(0, x+y)	Pour all water from the 4 gallon jug to the 3 gallon jug

The listed production rules contain all the actions that could be performed by the agent in transferring the contents of jugs. But, to solve the water jug problem in a minimum number of moves, following set of rules in the given sequence should be performed:

Solution of water jug problem according to the production rules

S.No.	**4 gallon jug contents**	**3 gallon jug contents**	**Rule followed**
1.	0 gallon	0 gallon	Initial state
2.	0 gallon	3 gallons	Rule no.2
3.	3 gallons	0 gallon	Rule no. 9
4.	3 gallons	3 gallons	Rule no. 2
5.	4 gallons	2 gallons	Rule no. 7
6.	0 gallon	2 gallons	Rule no. 5
7.	2 gallons	0 gallon	Rule no. 9

On reaching the 7th attempt, we reach a state which is our goal state. Therefore, at this state, our problem is solved.

1. **Approach BFS**

Here, we keep exploring **all the different valid cases of the states of water in the jug simultaneously** until and unless we reach the required target water. As provided in the problem statement, at any given state we can do either of the following operations:

- Fill a jug
- Empty a jug
- Transfer water from one jug to another **until either of them gets completely filled or empty.**

2. **Approach Memoization:**
   An approach using BFS has been discussed in the previous para. In this para an approach using memoization and recursion is been discussed. At any point, there can be a total of six possibilities:
   - Empty the first jug completely
   - Empty the second jug completely
   - Fill the first jug
   - Fill the second jug
   - Fill the water from the second jug into the first jug until the first jug is full or the second jug has no water left
   - Fill the water from the first jug into the second jug until the second jug is full or the first jug has no water left
3. **Approach Recursion :**
   Using Recursion, visit all the six possible moves one by one until one of them returns True. Since there can be repetitions of same recursive calls, hence every return value is stored using memoization to avoid calling the recursive function again and returning the stored value.

Time Complexity: O(n*m).
Space Complexity: O(n*m). Where n and m are the quantity of jug1 and jug2, respectively.

ALGORITHM:

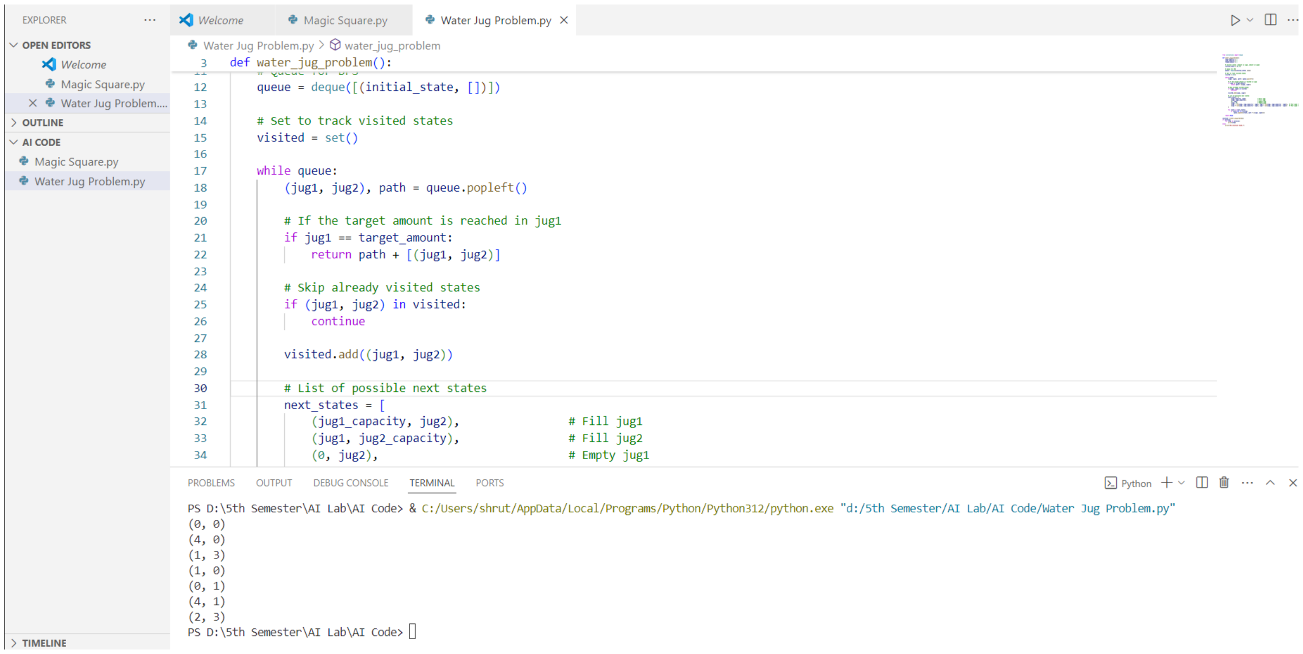
- Initialize two variables ***j1*** and ***j2*** to represent the current amount of water in each jug.
- Set a target amount of water to measure out.
- Create a list of possible actions, including filling a jug, emptying a jug, and pouring water from one jug to the other.
- Create an empty set to keep track of visited states.
- Create a stack to keep track of states to visit, and add the initial state to the stack.
- While the stack is not empty:
  - Pop the top state from the stack.
  - If this state has not been visited before, add it to the visited set.
  - Check if the state matches the target amount of water. If so, return the ***seq*** actions taken to get to this state.
  - If the state does not match the target amount, generate all possible next states from this state by applying each of the possible actions in turn.
  - If it has not been visited before, add it to the stack for each next state.
- If the stack becomes empty without finding a solution, return ***None***.

PROGRAM CODE:

```
from collections import deque

def water_jug_problem():
    jug1_capacity = 4
    jug2_capacity = 3
    target_amount = 2

    # Initial state: (amount in jug1, amount in jug2)
    initial_state = (0, 0)

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

    # Set to track visited states
    visited = set()

    while queue:
        (jug1, jug2), path = queue.popleft()

        # If the target amount is reached in jug1
        if jug1 == target_amount:
            return path + [(jug1, jug2)]

        # Skip already visited states
        if (jug1, jug2) in visited:
            continue

        visited.add((jug1, jug2))

        # List of possible next states
        next_states = [
            (jug1_capacity, jug2),              # Fill jug1
            (jug1, jug2_capacity),              # Fill jug2
            (0, jug2),                          # Empty jug1
            (jug1, 0),                          # Empty jug2
            (jug1 - min(jug1, jug2_capacity - jug2), jug2 + min(jug1, jug2_capacity - jug2)),  # Pour
jug1 to jug2
            (jug1 + min(jug2, jug1_capacity - jug1), jug2 - min(jug2, jug1_capacity - jug1))   # Pour
jug2 to jug1
        ]

        for state in next_states:
            if state not in visited:
                queue.append((state, path + [(jug1, jug2)]))

    return None
```

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

INPUT & OUTPUT:



```
def water_jug_problem():
    queue = deque([(initial_state, [])])

    # Set to track visited states
    visited = set()

    while queue:
        (jug1, jug2), path = queue.popleft()

        # If the target amount is reached in jug1
        if jug1 == target_amount:
            return path + [(jug1, jug2)]

        # Skip already visited states
        if (jug1, jug2) in visited:
            continue

        visited.add((jug1, jug2))

        # List of possible next states
        next_states = [
            (jug1_capacity, jug2),              # Fill jug1
            (jug1, jug2_capacity),              # Fill jug2
            (0, jug2),                          # Empty jug1
```

```
PS D:\5th Semester\AI Lab\AI Code> & C:/Users/shrut/AppData/Local/Programs/Python/Python312/python.exe "d:/5th Semester/AI Lab/AI Code/Water Jug Problem.py"
(0, 0)
(4, 0)
(1, 3)
(1, 0)
(0, 1)
(4, 1)
(2, 3)
PS D:\5th Semester\AI Lab\AI Code> 
```

CONCLUSION:

I successfully apply the concept of production rules to solve the water jug problem.

DISCUSSION QUESTIONS:

Which Technique is best for water jug problem?

Give second solution for water jug problem and write production rules.

REFERENCES:

https://www.geeksforgeeks.org/water-jug-problem-using-bfs/

https://www.geeksforgeeks.org/water-jug-problem-using-memoization/

https://favtutor.com/blogs/water-jug-problem

https://www.codingninjas.com/studio/library/solving-water-jug-problem-using-bfs

https://www.javatpoint.com/water-jug-problem-in-python