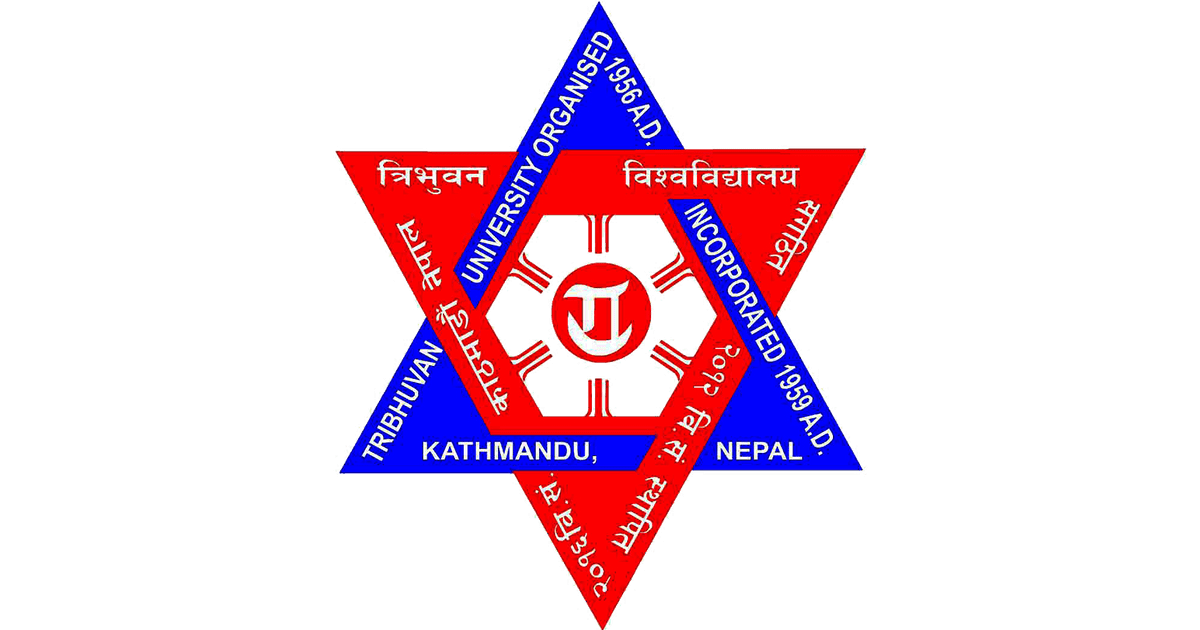
|  |  |
| --- | --- |
| Student ID: | 23081024 |
| Student Name: | Deepan Paudel |
| Course Name: | Artificial Intelligence |
| Course Code: | CSC261 |
| Semester: | 4th |
| Lab No.: | 05 |
| Date of Submission: | 14th August 2024 |
| Instructor/Examiner:  Evaluator’s Comment: | Mr. Saroj Maharjan |
| Evaluator’s Signature: | …..………………. |

LAB REPORT COVER PAGE



1. WAP to solve water jug problem.

OBJECTIVE

The objective of the Water Jug Problem is to determine if a specific amount of water can be measured using two jugs with given capacities and, if possible, find the sequence of operations to achieve this.

CODE:

def water\_jug\_problem(capacity1, capacity2, target):

def dfs(jug1, jug2, path):

# Check if we have reached the target

if jug1 == target or jug2 == target:

print("It is possible to measure the target amount. Steps:")

for step in path:

print(step)

return True

# If the state has already been visited, return False

if (jug1, jug2) in visited:

return False

# Mark the state as visited

visited.add((jug1, jug2))

# Try all possible actions

actions = [

(capacity1, jug2, f"Fill jug1: ({capacity1}, {jug2})"),

(jug1, capacity2, f"Fill jug2: ({jug1}, {capacity2})"),

(0, jug2, f"Empty jug1: (0, {jug2})"),

(jug1, 0, f"Empty jug2: ({jug1}, 0)"),

(jug1 - min(jug1, capacity2 - jug2), jug2 + min(jug1, capacity2 - jug2),

f"Pour jug1 to jug2: ({jug1 - min(jug1, capacity2 - jug2)}, {jug2 + min(jug1, capacity2 - jug2)})"),

(jug1 + min(jug2, capacity1 - jug1), jug2 - min(jug2, capacity1 - jug1),

f"Pour jug2 to jug1: ({jug1 + min(jug2, capacity1 - jug1)}, {jug2 - min(jug2, capacity1 - jug1)})")

]

# Recursively explore each action

for new\_jug1, new\_jug2, action in actions:

if dfs(new\_jug1, new\_jug2, path + [action]):

return True

return False

visited = set()

if dfs(0, 0, []):

return True

else:

print("It is not possible to measure the target amount.")

return False

# Example usage

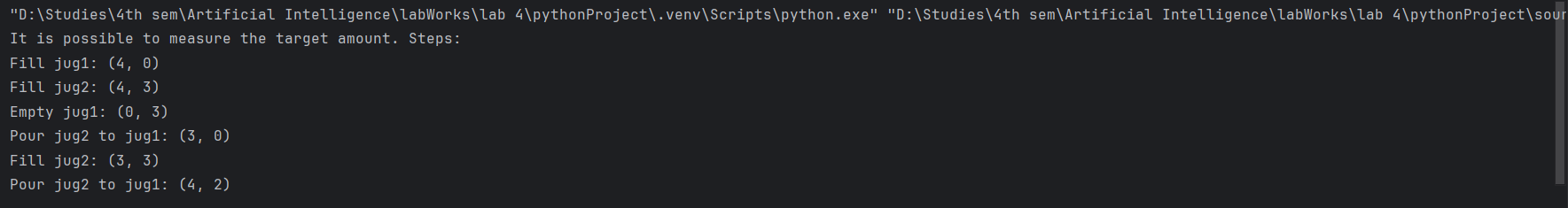
capacity1 = 4 # Capacity of jug 1

capacity2 = 3 # Capacity of jug 2

target = 2 # Target amount of water

water\_jug\_problem(capacity1, capacity2, target)

OUTPUT



CONCLUSION

The Water Jug Problem can be effectively solved using a breadth-first search (BFS) approach. This method systematically explores all possible states of the jugs and the operations that transition between these states, ensuring that no possible solution is overlooked. By keeping track of visited states and the sequence of operations, BFS can determine whether it is possible to achieve the desired amount of water and provide a step-by-step sequence of operations to do so. The algorithm demonstrates practical applications of search strategies and state exploration in problem-solving scenarios.