Fr. Conceicao Rodrigues College of Engineering
Fr. Agnel Ashram, Bandstand, Bandra (W),
Mumbai - 400050

Department of Computer Engineering
Academic Term II: 23-24

Class: B.E (Computer), Sem – VI Subject Name: Artificial Intelligence Student

Name: Sumit Sanjay Rai **Roll No: 9570**

Practical No:	**3**
Title:	Use DFS problem solving method for a) Water Jug Problem b) Missionaries & Cannibals
Date of Performance:	**12/02/2024**
Date of Submission:	**19/02/2024**

Rubrics for Evaluation:

Sr. No	**Performance Indicator**	**Excellent**	**Good**	**Below Average**	**Marks**
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Logic/Algorithm Complexity analysis (03)	03(Correct)	02(Partial)	01 (Tried)	
3	Coding Standards (03): Comments/indention/Naming conventions Test Cases /Output	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Assignment (03)	03(done well)	2 (Partially Correct)	1(submitted)	
Total					

Signature of the Teacher:

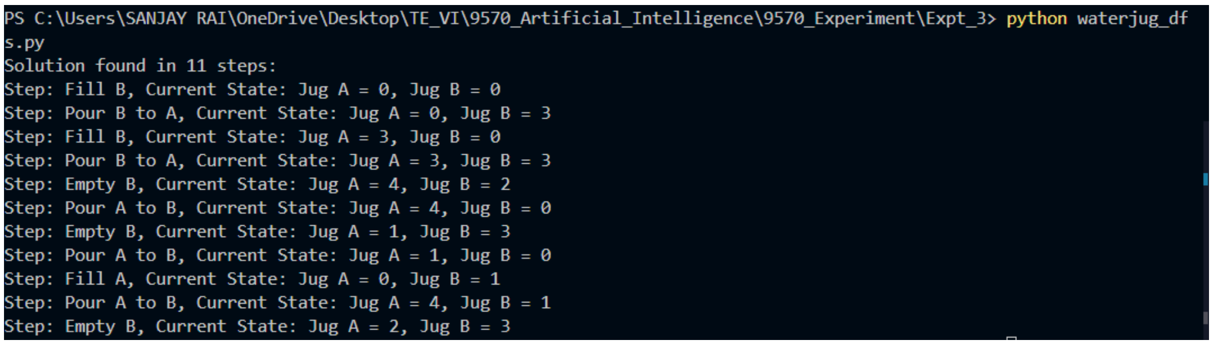
a) Water Jug Problem:

Source code:

```
def dfs_water_jug(capacity_a, capacity_b, target):
    stack = [(0, 0, [])]  # (current state A, current state B, path)
    visited = set()

    while stack:
        current_state_a, current_state_b, path = stack.pop()

        if (current_state_a, current_state_b) == target:
            return path

        if (current_state_a, current_state_b) in visited:
            continue

        visited.add((current_state_a, current_state_b))

        # Fill jug A
        stack.append((capacity_a, current_state_b, path + [(current_state_a, current_state_b, 'Fill A')]))

        # Fill jug B
        stack.append((current_state_a, capacity_b, path + [(current_state_a, current_state_b, 'Fill B')]))

        # Empty jug A
        stack.append((0, current_state_b, path + [(current_state_a, current_state_b, 'Empty A')]))

        # Empty jug B
        stack.append((current_state_a, 0, path + [(current_state_a, current_state_b, 'Empty B')]))

        # Pour water from jug A to jug B
        pour_amount = min(current_state_a, capacity_b - current_state_b)
        stack.append((current_state_a - pour_amount, current_state_b + pour_amount,
                      path + [(current_state_a, current_state_b, 'Pour A to B')]))

        # Pour water from jug B to jug A
        pour_amount = min(current_state_b, capacity_a - current_state_a)
        stack.append((current_state_a + pour_amount, current_state_b - pour_amount,
                      path + [(current_state_a, current_state_b, 'Pour B to A')]))

    return None  # No solution found


# Example usage:
capacity_a = 4
capacity_b = 3
target_amount = (2, 0)

result = dfs_water_jug(capacity_a, capacity_b, target_amount)
```

```
if result:
    print(f"Solution found in {len(result)} steps:")
    for step in result:
        print(f"Step: {step[-1]}, Current State: Jug A = {step[0]}, Jug B = {step[1]}")
else:
    print("No solution found.")
```

Output:



```
PS C:\Users\SANJAY RAI\OneDrive\Desktop\TE_VI\9570_Artificial_Intelligence\9570_Experiment\Expt_3> python waterjug_df
s.py
Solution found in 11 steps:
Step: Fill B, Current State: Jug A = 0, Jug B = 0
Step: Pour B to A, Current State: Jug A = 0, Jug B = 3
Step: Fill B, Current State: Jug A = 3, Jug B = 0
Step: Pour B to A, Current State: Jug A = 3, Jug B = 3
Step: Empty B, Current State: Jug A = 4, Jug B = 2
Step: Pour A to B, Current State: Jug A = 4, Jug B = 0
Step: Empty B, Current State: Jug A = 1, Jug B = 3
Step: Pour A to B, Current State: Jug A = 1, Jug B = 0
Step: Fill A, Current State: Jug A = 0, Jug B = 1
Step: Pour A to B, Current State: Jug A = 4, Jug B = 1
Step: Empty B, Current State: Jug A = 2, Jug B = 3
```

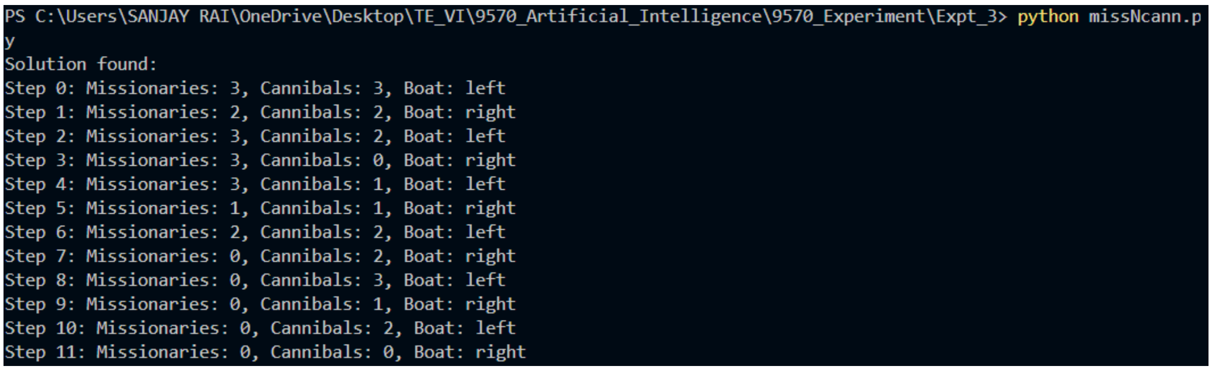
b) Missionaries & Cannibals:

Source code:

```
class State:
    def __init__(self, missionaries, cannibals, boat):
        self.missionaries = missionaries
        self.cannibals = cannibals
        self.boat = boat

    def is_valid(self):
        if self.missionaries < 0 or self.cannibals < 0 or self.missionaries > 3 or self.cannibals > 3:
            return False
        if self.missionaries < self.cannibals and self.missionaries > 0:
            return False
        if (3 - self.missionaries) < (3 - self.cannibals) and (3 - self.missionaries) > 0:
            return False
        return True

    def is_goal(self):
        return self.missionaries == 0 and self.cannibals == 0 and self.boat == 0

    def __eq__(self, other):
        return self.missionaries == other.missionaries and self.cannibals == other.cannibals and self.boat
== other.boat

    def __hash__(self):
        return hash((self.missionaries, self.cannibals, self.boat))

    def __repr__(self):
        return f"Missionaries: {self.missionaries}, Cannibals: {self.cannibals}, Boat: {'left' if self.boat == 1
else 'right'}"
```

```
# Actions represented using vector subtraction/addition
ACTIONS = [(1, 0, 1), (2, 0, 1), (0, 1, 1), (0, 2, 1), (1, 1, 1)]


def successors(state):
    moves = []
    for action in ACTIONS:
        if state.boat == 1:
            new_state = State(state.missionaries - action[0], state.cannibals - action[1], 0)
        else:
            new_state = State(state.missionaries + action[0], state.cannibals + action[1], 1)
        if new_state.is_valid():
            moves.append(new_state)
    return moves


def dfs(start_state, visited):
    stack = [(start_state, [start_state])]
    while stack:
        (state, path) = stack.pop()
        if state.is_goal():
            return path
        if state not in visited:
            visited.add(state)
            for successor in successors(state):
                if successor not in visited:
                    stack.append((successor, path + [successor]))
    return None


def print_solution(solution):
    for i, state in enumerate(solution):
        print(f"Step {i}: {state}")


def main():
    initial_state = State(3, 3, 1)
    visited = set()
    solution = dfs(initial_state, visited)
    if solution:
        print("Solution found:")
        print_solution(solution)
    else:
        print("No solution found.")


if __name__ == "__main__":
    main()
```

Output:



```
PS C:\Users\SANJAY RAI\OneDrive\Desktop\TE_VI\9570_Artificial_Intelligence\9570_Experiment\Expt_3> python missNcann.p
y
Solution found:
Step 0: Missionaries: 3, Cannibals: 3, Boat: left
Step 1: Missionaries: 2, Cannibals: 2, Boat: right
Step 2: Missionaries: 3, Cannibals: 2, Boat: left
Step 3: Missionaries: 3, Cannibals: 0, Boat: right
Step 4: Missionaries: 3, Cannibals: 1, Boat: left
Step 5: Missionaries: 1, Cannibals: 1, Boat: right
Step 6: Missionaries: 2, Cannibals: 2, Boat: left
Step 7: Missionaries: 0, Cannibals: 2, Boat: right
Step 8: Missionaries: 0, Cannibals: 3, Boat: left
Step 9: Missionaries: 0, Cannibals: 1, Boat: right
Step 10: Missionaries: 0, Cannibals: 2, Boat: left
Step 11: Missionaries: 0, Cannibals: 0, Boat: right
```