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:	6
Title:	Implementation of AO* algorithm
Date of Performance:	**11/03/2024**
Date of Submission:	**18/03/2024**

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					

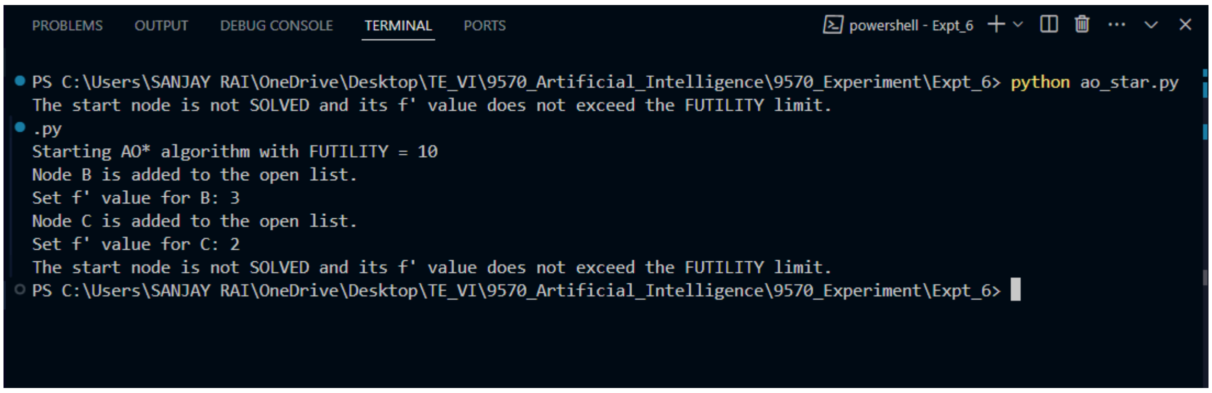
Signature of the Teacher:

Source Code:

```
class Node:
    def __init__(self, name):
        self.name = name
        self.successors = {}
        self.solved = False
        self.f_prime = None

    def add_successor(self, node, cost):
        self.successors[node] = cost

    def is_solved(self):
        return self.solved

    def mark_solved(self):
        self.solved = True

    def set_f_prime(self, f_prime):
        self.f_prime = f_prime

    def get_f_prime(self):
        return self.f_prime


def ao_star_search(start_node, f_utility):
    open_list = [start_node]

    while open_list:
        current_node = open_list.pop(0)

        if current_node.is_solved() or current_node.get_f_prime() > f_utility:
            continue

        if not current_node.successors:
            current_node.mark_solved()
            update_f_prime(current_node)
            print(f"Node {current_node.name} is marked as SOLVED.")
            print(f"Updated f' value for {current_node.name}: {current_node.get_f_prime()}")
            continue

        for successor, cost in current_node.successors.items():
            if successor.is_solved():
                current_node.mark_solved()
                update_f_prime(current_node)
                print(f"Node {current_node.name} is marked as SOLVED.")
            print(f"Updated f' value for {current_node.name}: {current_node.get_f_prime()}")
            break
        else:
            successor_f_prime = calculate_f_prime(successor)
            if successor_f_prime <= f_utility:
                open_list.append(successor)
                successor.set_f_prime(successor_f_prime)
                print(f"Node {successor.name} is added to the open list.")
                print(f"Set f' value for {successor.name}: {successor.get_f_prime()}")

    return start_node.is_solved() or start_node.get_f_prime() > f_utility


def calculate_f_prime(node):
    min_f_prime = float('inf')
    for successor, cost in node.successors.items():
        if successor.is_solved():
            f_prime = cost
        else:
            f_prime = cost + successor.get_f_prime()
        min_f_prime = min(min_f_prime, f_prime)
    return min_f_prime


def update_f_prime(node):
    for successor, cost in node.successors.items():
        if not successor.is_solved():
            successor.set_f_prime(calculate_f_prime(successor))


# Example usage:
if __name__ == "__main__":
    # Creating nodes
    A = Node('A')
    B = Node('B')
    C = Node('C')
    D = Node('D')

    # Adding successors
    A.add_successor(B, 5)
    A.add_successor(C, 7)
    B.add_successor(D, 3)
    C.add_successor(D, 2)

    # Setting f' for initial nodes
```

```
A.set_f_prime(0)
B.set_f_prime(0)
C.set_f_prime(0)
D.set_f_prime(0)

# Running AO* algorithm
f_utility = 10
print(f"Starting AO* algorithm with FUTILITY = {f_utility}")
result = ao_star_search(A, f_utility)

if result:
  print("The start node is SOLVED or its f' value exceeds the FUTILITY limit.")
else:
  print("The start node is not SOLVED and its f' value does not exceed the FUTILITY limit.")
```

Output:



```
PS C:\Users\SANJAY RAI\OneDrive\Desktop\TE_VI\9570_Artificial_Intelligence\9570_Experiment\Expt_6> python ao_star.py
The start node is not SOLVED and its f' value does not exceed the FUTILITY limit.
.py
Starting AO* algorithm with FUTILITY = 10
Node B is added to the open list.
Set f' value for B: 3
Node C is added to the open list.
Set f' value for C: 2
The start node is not SOLVED and its f' value does not exceed the FUTILITY limit.
PS C:\Users\SANJAY RAI\OneDrive\Desktop\TE_VI\9570_Artificial_Intelligence\9570_Experiment\Expt_6>
```