AI Experiment-6

April 17, 2024

[ ]:
```
from queue import PriorityQueue

class Node:
    def __init__(self, state, g_value, f_value):
        self.state = state
        self.g_value = g_value  # Actual cost from the start node
        self.f_value = f_value  # Optimistic value based on the heuristic

    def __lt__(self, other):
        # Comparing nodes based on their optimistic f_value
        return self.f_value < other.f_value

def ao_star_search(initial_state, goal_test, successors, heuristic):
    frontier = PriorityQueue()
    explored = set()

    # Initialize the start node
    start_node = Node(initial_state, 0, heuristic(initial_state))
    frontier.put(start_node)

    while not frontier.empty():
        current_node = frontier.get()

        if goal_test(current_node.state):
            return current_node.state

        explored.add(current_node.state)

        for successor in successors(current_node.state):
            successor_g_value = current_node.g_value + 1  # Assuming uniform␣
 ↪cost
            successor_f_value = successor_g_value + heuristic(successor)

            # If the successor state is not in the explored set, add it to the␣
 ↪frontier
            if successor not in explored:
```

```
                frontier.put(Node(successor, successor_g_value,␣
 ↪successor_f_value))

    return None

# Example usage:
def goal_test(state):
    return state == (4, 4)  # Example goal state

def successors(state):
    x, y = state
    return [(x+1, y), (x-1, y), (x, y+1), (x, y-1)]  # Example successors

def heuristic(state):
    # Manhattan distance heuristic
    x, y = state
    return abs(3 - x) + abs(3 - y)  # Assuming the goal state is (3, 3)

initial_state = (0, 0)  # Example initial state
result = ao_star_search(initial_state, goal_test, successors, heuristic)
print("Result:", result)
```

```
Result: (4, 4)
```