AI LAB 5 – 9763-Harsh Parmar– Batch D

Eight puzzle game solution by A* algorithm:

CODE:

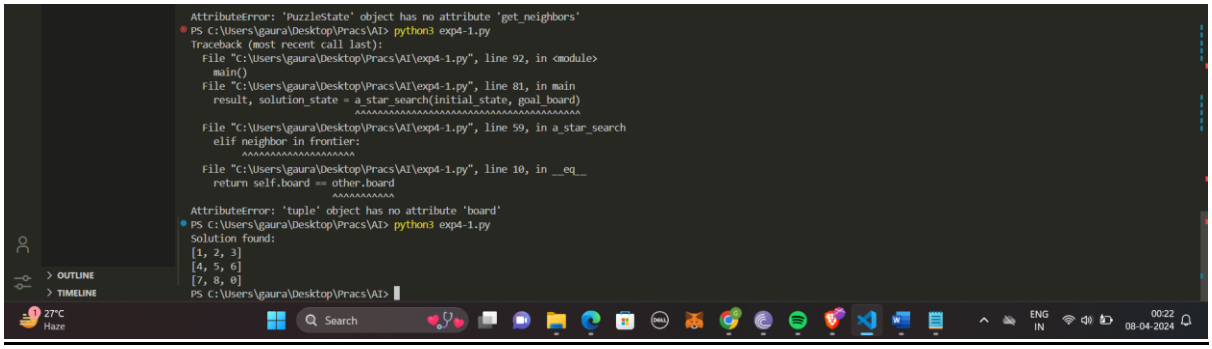
```
from heapq import heappush, heappop

# Define a class for the puzzle state
class PuzzleState:
    def __init__(self, board, goal):
        self.board = board
        self.goal = goal

    def __eq__(self, other):
        if isinstance(other, PuzzleState):
            return self.board == other.board
        return False

    def __hash__(self):
        return hash(str(self.board))

    # Define a method to calculate the heuristic (Manhattan distance)
    def heuristic(self):
        distance = 0
        for i in range(3):
```

```
            for j in range(3):
                if self.board[i][j] != self.goal[i][j]:
                    distance += 1
        return distance

    # Define a method to get neighboring states
    def get_neighbors(self):
        neighbors = []
        blank_i, blank_j = self.find_blank()
        moves = [(0, 1), (1, 0), (0, -1), (-1, 0)]  # Possible moves: right, down, left, up
        for move in moves:
            new_i, new_j = blank_i + move[0], blank_j + move[1]
            if 0 <= new_i < 3 and 0 <= new_j < 3:
                new_board = [row[:] for row in self.board]
                new_board[blank_i][blank_j], new_board[new_i][new_j] = new_board[new_i][new_j], new_board[blank_i][blank_j]
                neighbors.append(PuzzleState(new_board, self.goal))
        return neighbors

    # Define a method to find the position of the blank tile
    def find_blank(self):
        for i in range(3):
            for j in range(3):
                if self.board[i][j] == 0:
                    return i, j
```

```
# Define the A* search function
def a_star_search(initial_state, goal_state):
    frontier = []
    explored = set()
    heappush(frontier, (initial_state.heuristic(), 0, initial_state))  # (f(n), g(n), state)

    while frontier:
        _, cost, state = heappop(frontier)
        explored.add(state)
        if state.board == goal_state:
            return "SUCCESS", state

        for neighbor in state.get_neighbors():
            if neighbor not in explored.union(set(frontier)):
                heappush(frontier, (cost + neighbor.heuristic(), cost + 1, neighbor))
            elif neighbor in frontier:
                for idx, (f, g, s) in enumerate(frontier):
                    if s == neighbor and cost + 1 < g:
                        frontier[idx] = (f, cost + 1, neighbor)
                        break

    return "FAILURE", None

# Main function to test the A* algorithm
def main():
    # Define the initial and goal states
```

```
    initial_board = [[1, 2, 3],
                     [4, 5, 6],
                     [0, 7, 8]]
    goal_board = [[1, 2, 3],
                  [4, 5, 6],
                  [7, 8, 0]]

    # Create initial and goal puzzle states
    initial_state = PuzzleState(initial_board, goal_board)

    # Run A* search
    result, solution_state = a_star_search(initial_state, goal_board)

    # Print the result
    if result == "SUCCESS":
        print("Solution found:")
        for row in solution_state.board:
            print(row)
    else:
        print("No solution found.")

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

OUTPUT:


AttributeError: 'PuzzleState' object has no attribute 'get_neighbors'
PS C:\Users\gaura\Desktop\Pracs\AI> python3 exp4-1.py
Traceback (most recent call last):
File "C:\Users\gaura\Desktop\Pracs\AI\exp4-1.py", line 92, in <module>
main()
File "C:\Users\gaura\Desktop\Pracs\AI\exp4-1.py", line 81, in main
result, solution_state = a_star_search(initial_state, goal_board)
File "C:\Users\gaura\Desktop\Pracs\AI\exp4-1.py", line 59, in a_star_search
elif neighbor in frontier:
File "C:\Users\gaura\Desktop\Pracs\AI\exp4-1.py", line 10, in __eq__
return self.board == other.board
AttributeError: 'tuple' object has no attribute 'board'
PS C:\Users\gaura\Desktop\Pracs\AI> python3 exp4-1.py
Solution found:
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]
PS C:\Users\gaura\Desktop\Pracs\AI>
OUTLINE
TIMELINE
27°C
Haze
Search
ENG
IN
00:22
08-04-2024