A* Search Algorithm

```
import heapq

class PuzzleState:
    def __init__(self, board, depth=0, path=''):
        self.board = board
        self.blank_index = board.index(0)  # Index of the blank tile
        self.depth = depth
        self.path = path  # Path taken to reach this state

    def is_goal(self):
        return self.board == [1, 2, 3, 4, 5, 6, 7, 8, 0]

    def get_possible_moves(self):
        moves = []
        row, col = divmod(self.blank_index, 3)

        if row > 0:  # Up
            moves.append(-3)
        if row < 2:  # Down
            moves.append(3)
        if col > 0:  # Left
            moves.append(-1)
        if col < 2:  # Right
            moves.append(1)

        return moves

    def generate_new_state(self, move):
        new_index = self.blank_index + move
        new_board = self.board[:]
        new_board[self.blank_index], new_board[new_index] =
new_board[new_index], new_board[self.blank_index]
        return PuzzleState(new_board, self.depth + 1, self.path +
str(new_board))

    def heuristic_misplaced(self):
        return sum(1 for i in range(9) if self.board[i] != 0 and
self.board[i] != i + 1)
```

```
    def heuristic_manhattan(self):
        distance = 0
        for i in range(9):
            if self.board[i] != 0:
                target_row, target_col = divmod(self.board[i] - 1, 3)
                current_row, current_col = divmod(i, 3)
                distance += abs(target_row - current_row) + abs(target_col
- current_col)
        return distance

    def __lt__(self, other):
        return False  # Needed for priority queue to compare states


def a_star(initial_board, heuristic_type='misplaced'):
    start_state = PuzzleState(initial_board)
    if start_state.is_goal():
        return start_state.path

    # Priority queue for open states
    open_set = []
    heapq.heappush(open_set, (0, start_state))

    # Set to track visited states
    visited = set()

    while open_set:
        current_cost, current_state = heapq.heappop(open_set)

        if current_state.is_goal():
            return current_state.path

        visited.add(tuple(current_state.board))

        for move in current_state.get_possible_moves():
            new_state = current_state.generate_new_state(move)
            if tuple(new_state.board) in visited:
                continue

            # Calculate h(n) based on selected heuristic
```

```
            if heuristic_type == 'misplaced':
                h = new_state.heuristic_misplaced()
            elif heuristic_type == 'manhattan':
                h = new_state.heuristic_manhattan()
            else:
                raise ValueError("Invalid heuristic type")

            # Calculate f(n)
            f = new_state.depth + h

            # Print f(n) for the current state
            print(f"Current State: {new_state.board}, g(n):
{new_state.depth}, h(n): {h}, f(n): {f}")

            heapq.heappush(open_set, (f, new_state))

    return None  # No solution found


# Example usage
initial_board = [1, 2, 3, 4, 5, 6, 0, 7, 8]  # 0 represents the blank tile
print("Solution Path (Misplaced Tiles):", a_star(initial_board,
heuristic_type='misplaced'))
print("Solution Path (Manhattan Distance):", a_star(initial_board,
heuristic_type='manhattan'))

# Author Information
name = "Varsha Prasanth"
usn = "1BM22CS321"
print(f"Name: {name}, USN: {usn}")
```

Output

```
Current State: [1, 2, 3, 0, 5, 6, 4, 7, 8], g(n): 1, h(n): 3, f(n): 4
Current State: [1, 2, 3, 4, 5, 6, 7, 0, 8], g(n): 1, h(n): 1, f(n): 2
Current State: [1, 2, 3, 4, 0, 6, 7, 5, 8], g(n): 2, h(n): 2, f(n): 4
Current State: [1, 2, 3, 4, 5, 6, 7, 8, 0], g(n): 2, h(n): 0, f(n): 2
Solution Path (Misplaced Tiles): [1, 2, 3, 4, 5, 6, 7, 0, 8][1, 2, 3, 4, 5, 6, 7, 8, 0]
Current State: [1, 2, 3, 0, 5, 6, 4, 7, 8], g(n): 1, h(n): 3, f(n): 4
Current State: [1, 2, 3, 4, 5, 6, 7, 0, 8], g(n): 1, h(n): 1, f(n): 2
Current State: [1, 2, 3, 4, 0, 6, 7, 5, 8], g(n): 2, h(n): 2, f(n): 4
Current State: [1, 2, 3, 4, 5, 6, 7, 8, 0], g(n): 2, h(n): 0, f(n): 2
Solution Path (Manhattan Distance): [1, 2, 3, 4, 5, 6, 7, 0, 8][1, 2, 3, 4, 5, 6, 7, 8, 0]
Name: Varsha Prasanth, USN: 1BM22CS321
```