DFS 8 PUZZLE SOLVE

CODE : print(f"Santhosh N (1BM23CS302)")

```
MAX_VISITED_DISPLAY = 10  # Max number of visited states to print during DFS
NUM_INTERMEDIATE_STATES = 3  # How many intermediate states to show in the solution path

def print_state(state):
    for row in state:
        print(' '.join(str(x) for x in row))
    print()

def is_goal(state, goal_state):
    return state == goal_state

def find_zero(state):
    for i in range(3):
        for j in range(3):
            if state[i][j] == 0:
                return i, j

def get_neighbors(state):
    neighbors = []
    x, y = find_zero(state)
    directions = [(1,0), (-1,0), (0,1), (0,-1)]
    for dx, dy in directions:
        new_x, new_y = x + dx, y + dy
        if 0 <= new_x < 3 and 0 <= new_y < 3:
            new_state = [row[:] for row in state]
            new_state[x][y], new_state[new_x][new_y] = new_state[new_x][new_y], new_state[x][y]
            neighbors.append(new_state)
    return neighbors

def is_solvable(state):
    flat = [num for row in state for num in row if num != 0]
    inv_count = 0
    for i in range(len(flat)):
        for j in range(i + 1, len(flat)):
            if flat[i] > flat[j]:
                inv_count += 1
    return inv_count % 2 == 0

def dfs(start_state, goal_state):
    stack = [(start_state, [start_state])]
    visited = set()
```

DFS 8 PUZZLE SOLVE

```
    visited.add(tuple(tuple(row) for row in start_state))

    visited_count = 0

    print("Starting DFS traversal...\n")

    while stack:
        current_state, path = stack.pop()

        visited_count += 1
        if visited_count <= MAX_VISITED_DISPLAY:
            print(f"Visited state #{visited_count}:")
            print_state(current_state)

        if is_goal(current_state, goal_state):
            print(f"\nGoal reached!")
            print(f"Total visited states: {visited_count}")
            return path

        for neighbor in reversed(get_neighbors(current_state)):
            neighbor_tuple = tuple(tuple(row) for row in neighbor)
            if neighbor_tuple not in visited:
                visited.add(neighbor_tuple)
                stack.append((neighbor, path + [neighbor]))

    print(f"\nTotal visited states: {visited_count}")
    return None

def read_state(name):
    print(f"Enter the {name} state, row by row (use space-separated numbers, 0 for empty):")
    state = []
    for _ in range(3):
        row = input().strip().split()
        if len(row) != 3:
            raise ValueError("Each row must have exactly 3 numbers.")
        row = list(map(int, row))
        state.append(row)
    return state

# --- Main Execution ---

initial_state = read_state("initial")
goal_state = read_state("goal")
```

DFS 8 PUZZLE SOLVE

```
if not (is_solvable(initial_state) == is_solvable(goal_state)):
    print("The puzzle is unsolvable.")
    exit()

solution_path = dfs(initial_state, goal_state)

if solution_path:
    cost = len(solution_path) - 1
    print(f"\nSolution found with cost: {cost}\n")
    print("Solution path:")

    total_steps = len(solution_path) - 1  # number of moves
    print("Initial State:")
    print_state(solution_path[0])

    if total_steps > 1:
        step_indices = list(range(1, total_steps))
        if len(step_indices) > NUM_INTERMEDIATE_STATES:
            interval = len(step_indices) // (NUM_INTERMEDIATE_STATES + 1)
            selected_indices = [step_indices[i * interval] for i in range(1,
NUM_INTERMEDIATE_STATES + 1)]
        else:
            selected_indices = step_indices

        for idx in selected_indices:
            print(f"Intermediate State (Step {idx}):")
            print_state(solution_path[idx])

    print("Final State:")
    print_state(solution_path[-1])
else:
    print("No solution found")
```

DFS 8 PUZZLE SOLVE

OUTPUT :

```
Santhosh N (1BM23CS302)
Enter the initial state, row by row (use space-separated numbers, 0 for empty):
2 8 3
1 6 4
7 0 5
Enter the goal state, row by row (use space-separated numbers, 0 for empty):
1 2 3
8 0 4
7 6 5
Starting DFS traversal...

Visited state #1:
2 8 3
1 6 4
7 0 5

Visited state #2:
2 8 3
1 0 4
7 6 5

Visited state #3:
2 0 3
1 8 4
7 6 5

Visited state #4:
2 3 0
1 8 4
7 6 5

Visited state #5:
2 3 4
1 8 0
7 6 5

Visited state #6:
2 3 4
1 8 5
7 6 0

Visited state #7:
2 3 4
1 8 5
7 0 6

Visited state #8:
2 3 4
1 0 5
7 8 6

Visited state #9:
2 0 4
1 3 5
7 8 6

Visited state #10:
2 4 0
1 3 5
7 8 6
```

DFS 8 PUZZLE SOLVE

```
Visited state #9:
2 0 4
1 3 5
7 8 6

Visited state #10:
2 4 0
1 3 5
7 8 6


Goal reached!
Total visited states: 29317

Solution found with cost: 28013

Solution path:
Initial State:
2 8 3
1 6 4
7 0 5

Intermediate State (Step 7004):
8 0 1
6 3 4
5 7 2

Intermediate State (Step 14007):
6 3 0
7 1 8
4 2 5

Intermediate State (Step 21010):
8 4 3
1 7 5
6 0 2

Final State:
1 2 3
8 0 4
7 6 5
```