AI LAB 4

8 PUZZLE PROBLEM USING ITERATIVE DEEPENING DEPTH FIRST SEARCH

1BM21CS203

```
class PuzzleNode:
    def __init__(self, state, parent=None, action=None):
        self.state = state
        self.parent = parent
        self.action = action

    def get_path(self): # Not required
        path = []
        current = self
        while current:
            path.append((current.state, current.action))
            current = current.parent
        return path[::-1]

def is_goal(state):
    goal_state = (1,2,3,6,4,5,0,7,8)
    return state == goal_state

def get_neighbors(state):
    neighbors = []
    empty_index = state.index(0)
    row, col = divmod(empty_index, 3)

    for move in [(0, 1), (1, 0), (0, -1), (-1, 0)]:
        new_row, new_col = row + move[0], col + move[1]
        if 0 <= new_row < 3 and 0 <= new_col < 3:
            neighbor_state = list(state)
            neighbor_index = new_row * 3 + new_col
            neighbor_state[empty_index], neighbor_state[neighbor_index] = (
                neighbor_state[neighbor_index],
                neighbor_state[empty_index],
            )
            neighbors.append(tuple(neighbor_state))

    return neighbors

def depth_limited_search(node, goal_state, depth_limit):
    if is_goal(node.state):
        return True
    elif depth_limit == 0:
        return False
    else:
        for neighbor_state in get_neighbors(node.state):
            child = PuzzleNode(neighbor_state, node)
            if depth_limited_search(child, goal_state, depth_limit - 1):
```

```
                return True
        return False

if __name__ == "__main__":
    initial_state = (1, 2, 3, 0, 4, 5, 6, 7, 8)
    depth_limit = 1  # Set the depth limit as needed
    initial_node = PuzzleNode(initial_state)
    result = depth_limited_search(initial_node, (1,2,3,6,4,5,0,7,8),
depth_limit)
    print(result)
```

OUTPUT:

```
dfs.py
True
```