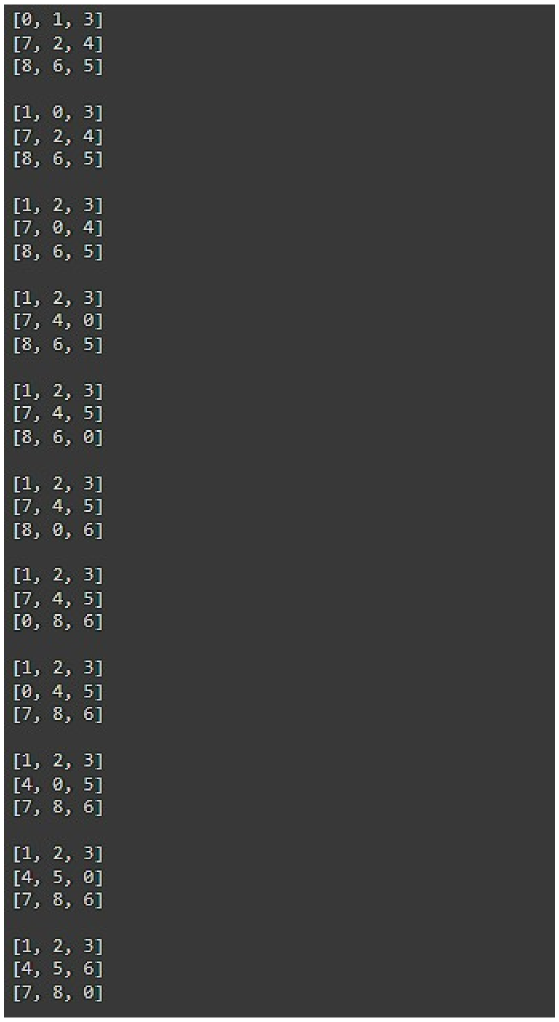
3. 8 puzzle DFS

```
def dfs(initial_board, zero_pos):
    stack = [(initial_board, zero_pos, [])]
    visited = set()

    while stack:
        current_board, zero_pos, moves = stack.pop()

        if is_goal(current_board):
            return moves, len(moves)  # Return moves and their count

        visited.add(tuple(current_board))

        for neighbor_board, neighbor_pos in get_neighbors(current_board, zero_pos):
            if tuple(neighbor_board) not in visited:
                stack.append((neighbor_board, neighbor_pos, moves + [neighbor_board]))

    return None, 0  # No solution found, return count as 0

# Initial state of the puzzle
initial_board = [1, 2, 3, 0, 4, 6, 7, 5, 8]
zero_position = (1, 0)  # Position of the empty tile (0)

# Solve the puzzle using DFS
```

```
solution, move_count = dfs(initial_board, zero_position)


if solution:
    print("Solution found with moves ({} moves):".format(move_count))
    for move in solution:
        print_board(move)
        print()  # Print an empty line between moves
else:
    print("No solution found.")
```

Truncated output


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

[1, 0, 3]
[7, 2, 4]
[8, 6, 5]

[1, 2, 3]
[7, 0, 4]
[8, 6, 5]

[1, 2, 3]
[7, 4, 0]
[8, 6, 5]

[1, 2, 3]
[7, 4, 5]
[8, 6, 0]

[1, 2, 3]
[7, 4, 5]
[8, 0, 6]

[1, 2, 3]
[7, 4, 5]
[0, 8, 6]

[1, 2, 3]
[0, 4, 5]
[7, 8, 6]

[1, 2, 3]
[4, 0, 5]
[7, 8, 6]

[1, 2, 3]
[4, 5, 0]
[7, 8, 6]

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