216. You and your friends are assigned the task of coloring a map with a limited number of colors. The map is represented as a list of regions and their adjacency relationships. The rules are as follows: At each step, you can choose any uncolored region and color it with any available color. Your friend Alice follows the same strategy immediately after you, and then your friend Bob follows suit. You want to maximize the number of regions you personally color. Write a function that takes the map's adjacency list representation and returns the maximum number of regions you can color before all regions are colored. Write a program to implement the Graph coloring technique for an undirected graph. Implement an algorithm with minimum number of colors. edges = [(0, 1), (1, 2), (2, 3), (3, 0), (0, 2)] No. of vertices, n = 4

PROGRAM:-

```
def graph_coloring(adj_list):
    colors = {}
    result = 0
    for node in sorted(adj_list, key=lambda x: len(adj_list[x]), reverse=True):
        neighbor_colors = set(colors.get(nei) for nei in adj_list[node])
        colors[node] = next(color for color in range(len(adj_list)) if color not in neighbor_colors)
        result = max(result, colors[node] + 1)
    return result

# Adjacency list representation of the graph
adj_list = {
    0: [1, 3, 2],
    1: [0, 2],
    2: [1, 0, 3],
    3: [2, 0]
}

# Calculate the maximum number of regions that can be colored
max_regions_colored = graph_coloring(adj_list)
print(max_regions_colored)
```

OUTPUT:-

TIME COMPLEXITY:- O(n+m),