DESIGN AND ANALYSIS OF ALGORITHM

PRACTICAL-8A

NAME: VEDANT BHUTADA

ROLL: 69

BATCH: A4

Aim: Implement Graph Colouring algorithm use Graph colouring concept.

Problem Statement: A GSM is a cellular network with its entire geographical range divided into hexadecimal cells. Each cell has a communication tower which connects with mobile phones within cell. Assume this GSM network operates in different frequency ranges. Allot frequencies to each cell such that no adjacent cells have same frequency range. Consider an undirected graph G = (V, E) shown in fig. Find the colour assigned to each node using Backtracking method. Input is the adjacency matrix of a graph G(V, E), where V is the number of Vertices and E is the number of edges.

```
def is_safe(vertex, color, graph, color_assigned):
    for i in range(len(graph)):
        if graph[vertex][i] == 1 and color_assigned[i] == color:
            return False
    return True

def graph_colouring_util(graph, num_of_vertices, num_of_colors, color_assigned, vertex, solutions):
    if vertex == num_of_vertices:
        # Print the current solution
        print("Vertex   Color")
        for i in range(num_of_vertices):
            print(f"  {i + 1}       {color_assigned[i]}")
        print("-------------")
        solutions[0] += 1
        return False  # Continue searching for more solutions

    for c in range(1, num_of_colors + 1):
        if is_safe(vertex, c, graph, color_assigned):
            color_assigned[vertex] = c
            if graph_colouring_util(graph, num_of_vertices, num_of_colors, color_assigned, vertex + 1, solutions):
                return True
            color_assigned[vertex] = 0  # Backtrack

    return False

def graph_colouring(graph, num_of_vertices, num_of_colors):
    color_assigned = [0] * num_of_vertices
    solutions = [0]  # Store the total number of solutions

    if not graph_colouring_util(graph, num_of_vertices, num_of_colors, color_assigned, 0, solutions):
        #print("No solution found")
        print(f"Total solutions found: {solutions[0]}")
    else:
        print(f"Total solutions found: {solutions[0]}")

# Example usage
adjacency_matrix = [
    [0, 1, 0, 1, 0],
    [1, 0, 1, 1, 1],
    [0, 1, 0, 0, 1],
    [1, 1, 0, 0, 1],
    [0, 1, 1, 1, 0]
]
num_of_vertices = 5
num_of_colors = 3
graph_colouring(adjacency_matrix, num_of_vertices, num_of_colors)
```

```
Vertex   Color
  1       1
  2       2
  3       3
  4       3
  5       1
-------------
Vertex   Color
  1       1
  2       3
  3       2
  4       2
  5       1
-------------
```

```
Vertex   Color
  1       2
  2       1
  3       3
  4       3
  5       2
-------------
Vertex   Color
  1       2
  2       3
  3       1
  4       1
  5       2
-------------
Vertex   Color
  1       3
  2       1
  3       2
  4       2
  5       3
-------------
Vertex   Color
  1       3
  2       2
  3       1
  4       1
  5       3
-------------
Total solutions found: 6
```