PRACTICAL 8

NEER AWASTHI

A4 B2 20  
  
**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.

**Source Code:**

**# Practical 8: Graph Coloring using Backtracking**

**# Problem: GSM network frequency allocation**

**# Function to check if it’s safe to assign color c to vertex v**

**def is\_safe(v, color, graph, c):**

**for i in range(len(graph)):**

**if graph[v][i] == 1 and color[i] == c:**

**return False**

**return True**

**# Function to solve graph coloring using backtracking**

**def graph\_coloring(graph, m, color, v):**

**# If all vertices are colored, return True**

**if v == len(graph):**

**return True**

**for c in range(1, m + 1):**

**if is\_safe(v, color, graph, c):**

**color[v] = c**

**if graph\_coloring(graph, m, color, v + 1):**

**return True**

**# Backtrack**

**color[v] = -1**

**return False**

**# Function to solve the graph coloring problem**

**def solve\_graph\_coloring(graph, m):**

**color = [-1] \* len(graph)**

**if not graph\_coloring(graph, m, color, 0):**

**print("Solution does not exist")**

**return False**

**print("Solution exists with the following coloring:")**

**for i in range(len(graph)):**

**print(f"Vertex {i} -> Color {color[i]}")**

**return True**

**# ---------------- Example Graphs ----------------**

**# Graph 1 (Adjacency Matrix)**

**graph1 = [**

**[0, 1, 1, 0],**

**[1, 0, 1, 1],**

**[1, 1, 0, 0],**

**[0, 1, 0, 0]**

**]**

**# Graph 2**

**graph2 = [**

**[0, 1, 0, 1],**

**[1, 0, 1, 0],**

**[0, 1, 0, 1],**

**[1, 0, 1, 0]**

**]**

**m = 3**

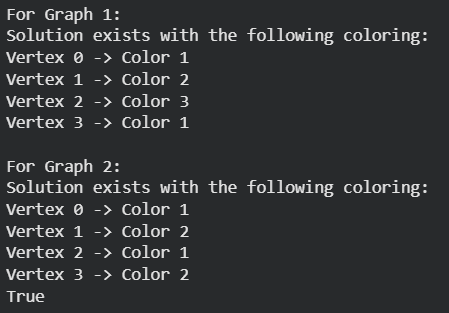
**print("For Graph 1:")**

**solve\_graph\_coloring(graph1, m)**

**print("\nFor Graph 2:")**

**solve\_graph\_coloring(graph2, m)**

**Output:**

****