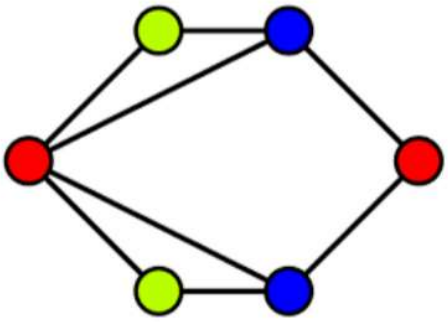
DAA Practical 8

Name	***Humanshu Rajesh Sakharkar***
Section / Roll no	***A4 B3 39***

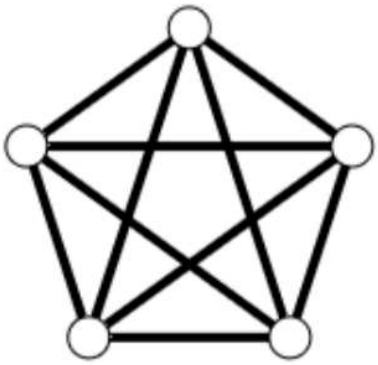
Aim *:- Implement Graph Colouring algorithm use Graph colouring concept.*

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.

Graph 1:



Graph 2:



Code :-

```
class GraphColoring:
    def __init__(self, graph, m):
        self.graph = graph
        self.V = len(graph)
        self.m = m
        self.color = [0] * self.V

        # Predefined color names for readability
        self.color_names = ["", "Red", "Green", "Blue", "Yellow",
"Orange", "Purple", "Pink", "Cyan"]

    def isSafe(self, v, c):
        """Check if assigning color c to vertex v is safe."""
        for i in range(self.V):
            if self.graph[v][i] == 1 and self.color[i] == c:
                return False
        return True

    def solve(self, v):
        """Recursive backtracking solver."""
        if v == self.V:
            return True

        for c in range(1, self.m + 1):
            if self.isSafe(v, c):
                self.color[v] = c
                if self.solve(v + 1):
                    return True
                self.color[v] = 0  # Backtrack
        return False

    def graphColoring(self):
        """Driver function to solve and display the coloring result."""
        if self.solve(0):
            print("Solution exists: Following are the assigned colors:")
            for i in range(self.V):
                color_name = self.color_names[self.color[i]] if
self.color[i] < len(self.color_names) else f"Color-{self.color[i]}"
                print(f"Vertex {i + 1} ---> {color_name}")
```

```
        else:
            print("No solution exists with the given number of colors.")


# ------------------------------------------------
print("-------------------------------------------------")
print("GRAPH 1: Coloring as per the image (given)")
print("-------------------------------------------------")

graph1_colors = {
    0: "Red",
    1: "Yellow-Green",
    2: "Blue",
    3: "Red",
    4: "Yellow-Green",
    5: "Blue"
}

for v, c in graph1_colors.items():
    print(f"Vertex {v + 1} ---> {c}")

# ------------------------------------------------
print("\n-------------------------------------------------")
print("GRAPH 2: Coloring using Backtracking Algorithm")
print("-------------------------------------------------")

graph2 = [
    [0, 1, 1, 1, 1],
    [1, 0, 1, 1, 1],
    [1, 1, 0, 1, 1],
    [1, 1, 1, 0, 1],
    [1, 1, 1, 1, 0]
]

g2 = GraphColoring(graph2, m=5)
g2.graphColoring()
```

Output :-

```
[Running] python -u "c:\Users\human\OneDrive\Desktop\Humanshu\pract8.py"
--------------------------------------------------
GRAPH 1: Coloring as per the image (given)
--------------------------------------------------
Vertex 1 ---> Red
Vertex 2 ---> Yellow-Green
Vertex 3 ---> Blue
Vertex 4 ---> Red
Vertex 5 ---> Yellow-Green
Vertex 6 ---> Blue

--------------------------------------------------
GRAPH 2: Coloring using Backtracking Algorithm
--------------------------------------------------
Solution exists: Following are the assigned colors:
Vertex 1 ---> Red
Vertex 2 ---> Green
Vertex 3 ---> Blue
Vertex 4 ---> Yellow
Vertex 5 ---> Orange

[Done] exited with code=0 in 0.127 seconds
```