Practical-8

Name: Krish Parothi

Section: A4

Batch: B3

Roll Number: 49

Aim: Implement Graph Colouring algorithm use Graph colouring concept.

Problem Statement:

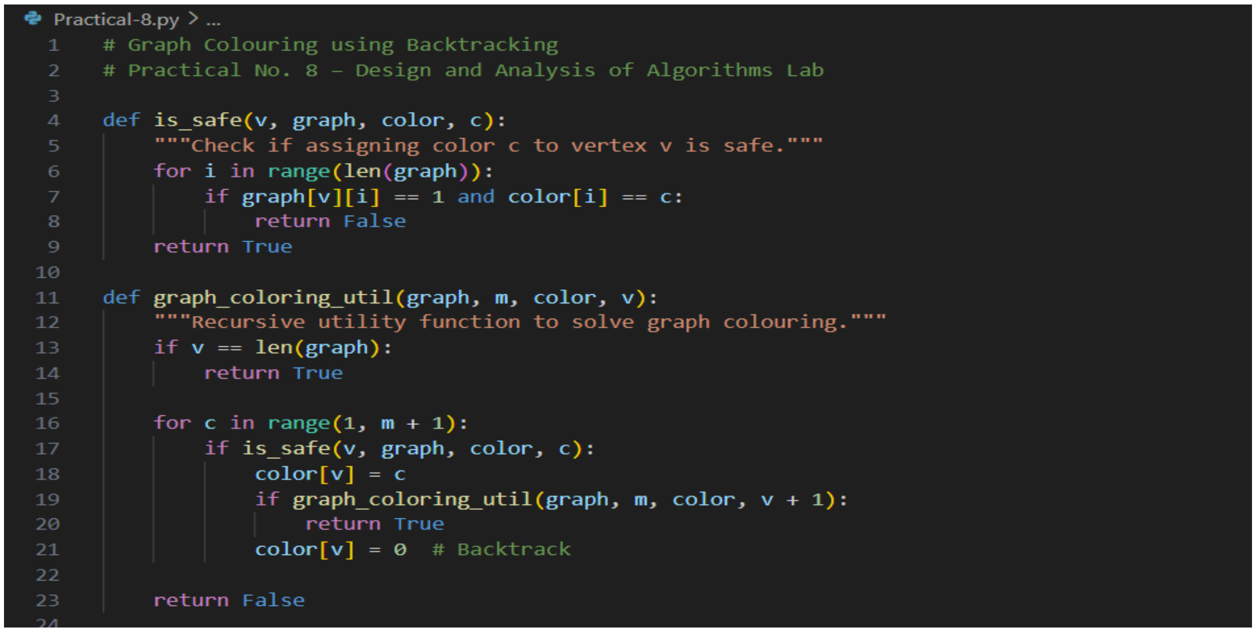
A GSM is a cellular network with its entire geographical range divided into hexadecimal cells. Each cell has a communication tower which connects with mobile phones within cell. Assume this GSM network operates in different frequency ranges. Allot frequencies to each cell such that no adjacent cells 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.

Code in Text Format:

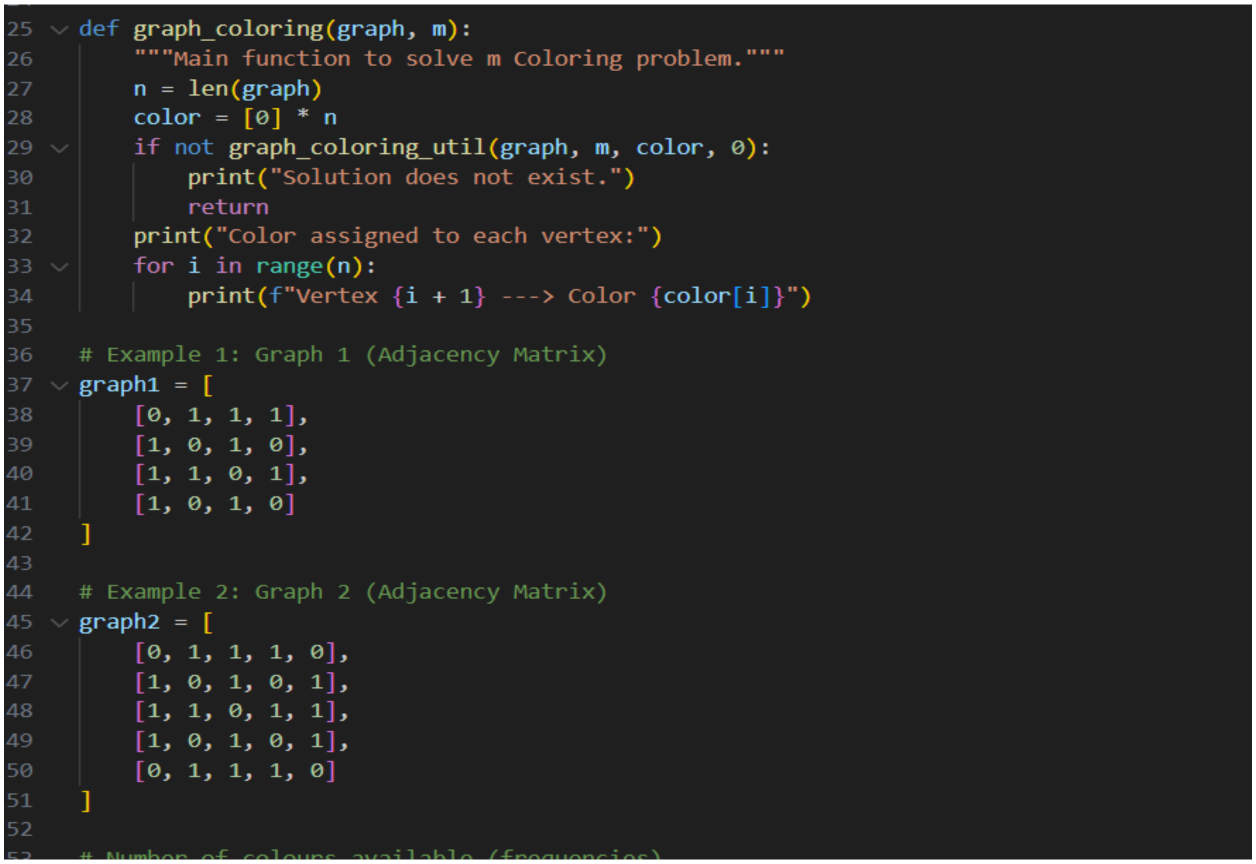
```
# Graph Colouring using Backtracking
# Practical No. 8 - Design and Analysis of Algorithms Lab

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

def graph_coloring_util(graph, m, color, v):
    """Recursive utility function to solve graph colouring."""
    if v == len(graph):
        return True

    for c in range(1, m + 1):
        if is_safe(v, graph, color, c):
            color[v] = c
            if graph_coloring_util(graph, m, color, v + 1):
                return True
            color[v] = 0  # Backtrack
    return False

def graph_coloring(graph, m):
    """Main function to solve m Coloring problem."""
    n = len(graph)
    color = [0] * n
    if not graph_coloring_util(graph, m, color, 0):
        print("Solution does not exist.")
        return
    print("Color assigned to each vertex:")
    for i in range(n):
        print(f"Vertex {i + 1} ---> Color {color[i]}")

# Example 1: Graph 1 (Adjacency Matrix)
graph1 = [
    [0, 1, 1, 1],
    [1, 0, 1, 0],
    [1, 1, 0, 1],
    [1, 0, 1, 0]
]

# Example 2: Graph 2 (Adjacency Matrix)
graph2 = [
    [0, 1, 1, 1, 0],
    [1, 0, 1, 0, 1],
    [1, 1, 0, 1, 1],
    [1, 0, 1, 0, 1],
    [0, 1, 1, 1, 0]
]

# Number of colours available (frequencies)
m = 3

print("Graph 1 Result:")
graph_coloring(graph1, m)

print("\nGraph 2 Result:")
graph_coloring(graph2, m)
```

Code Screenshot:

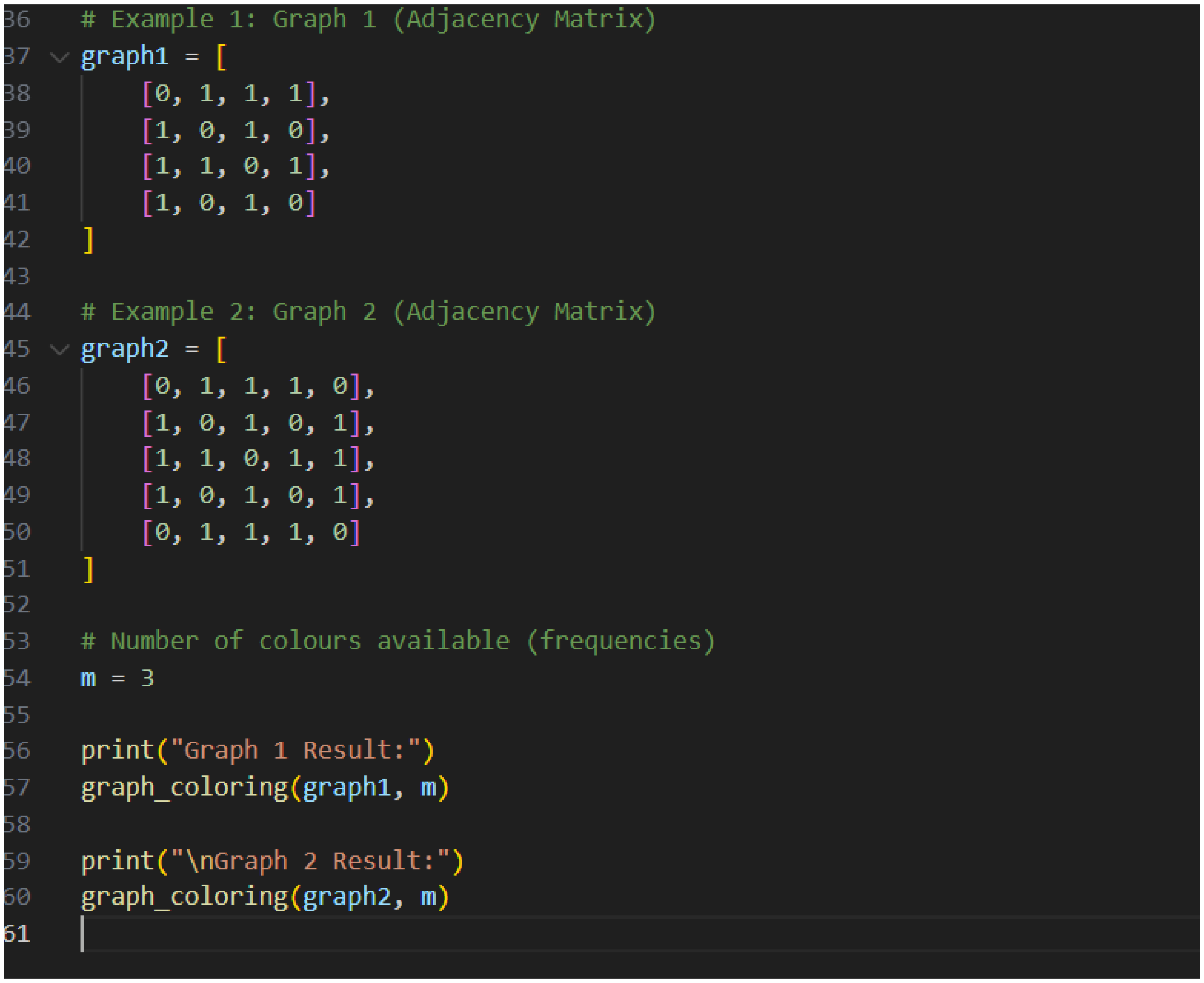


Practical-8.py > ...

```python
# Graph Colouring using Backtracking
# Practical No. 8 – Design and Analysis of Algorithms Lab

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

def graph_coloring_util(graph, m, color, v):
    """Recursive utility function to solve graph colouring."""
    if v == len(graph):
        return True

    for c in range(1, m + 1):
        if is_safe(v, graph, color, c):
            color[v] = c
            if graph_coloring_util(graph, m, color, v + 1):
                return True
            color[v] = 0  # Backtrack

    return False
```

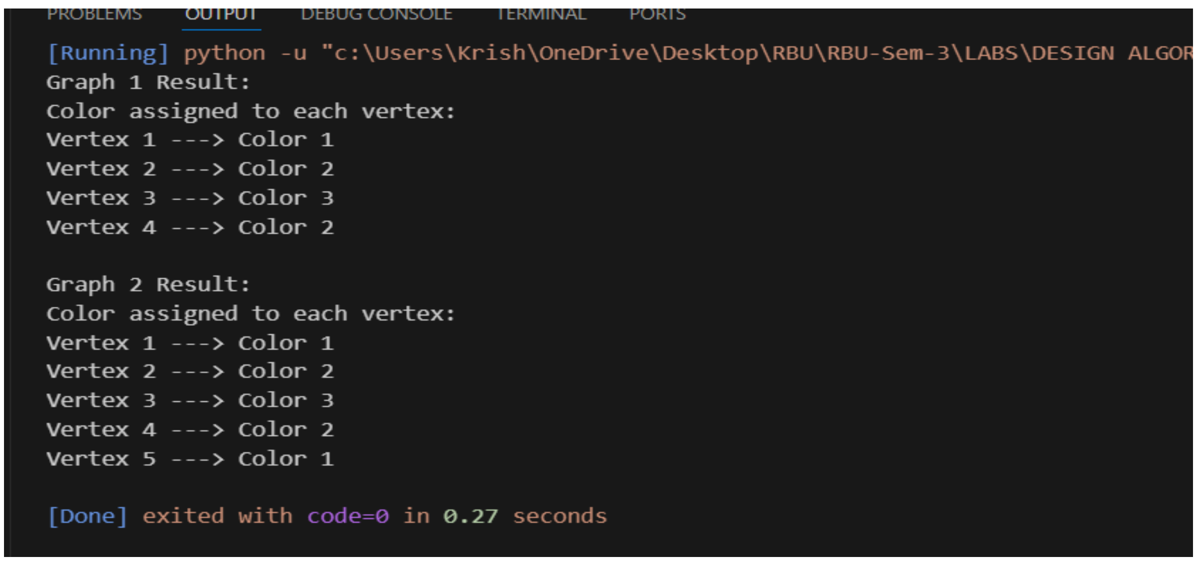


```python
def graph_coloring(graph, m):
    """Main function to solve m Coloring problem."""
    n = len(graph)
    color = [0] * n
    if not graph_coloring_util(graph, m, color, 0):
        print("Solution does not exist.")
        return
    print("Color assigned to each vertex:")
    for i in range(n):
        print(f"Vertex {i + 1} ---> Color {color[i]}")

# Example 1: Graph 1 (Adjacency Matrix)
graph1 = [
    [0, 1, 1, 1],
    [1, 0, 1, 0],
    [1, 1, 0, 1],
    [1, 0, 1, 0]
]

# Example 2: Graph 2 (Adjacency Matrix)
graph2 = [
    [0, 1, 1, 1, 0],
    [1, 0, 1, 0, 1],
    [1, 1, 0, 1, 1],
    [1, 0, 1, 0, 1],
    [0, 1, 1, 1, 0]
]

# Number of colours available (frequencies)
```



```python
# Example 1: Graph 1 (Adjacency Matrix)
graph1 = [
        [0, 1, 1, 1],
        [1, 0, 1, 0],
        [1, 1, 0, 1],
        [1, 0, 1, 0]
]

# Example 2: Graph 2 (Adjacency Matrix)
graph2 = [
        [0, 1, 1, 1, 0],
        [1, 0, 1, 0, 1],
        [1, 1, 0, 1, 1],
        [1, 0, 1, 0, 1],
        [0, 1, 1, 1, 0]
]

# Number of colours available (frequencies)
m = 3

print("Graph 1 Result:")
graph_coloring(graph1, m)

print("\nGraph 2 Result:")
graph_coloring(graph2, m)

```

Output:



```
[Running] python -u "c:\Users\Krish\OneDrive\Desktop\RBU\RBU-Sem-3\LABS\DESIGN ALGOR
Graph 1 Result:
Color assigned to each vertex:
Vertex 1 ---> Color 1
Vertex 2 ---> Color 2
Vertex 3 ---> Color 3
Vertex 4 ---> Color 2

Graph 2 Result:
Color assigned to each vertex:
Vertex 1 ---> Color 1
Vertex 2 ---> Color 2
Vertex 3 ---> Color 3
Vertex 4 ---> Color 2
Vertex 5 ---> Color 1

[Done] exited with code=0 in 0.27 seconds
```