Name: Sujal SINGH

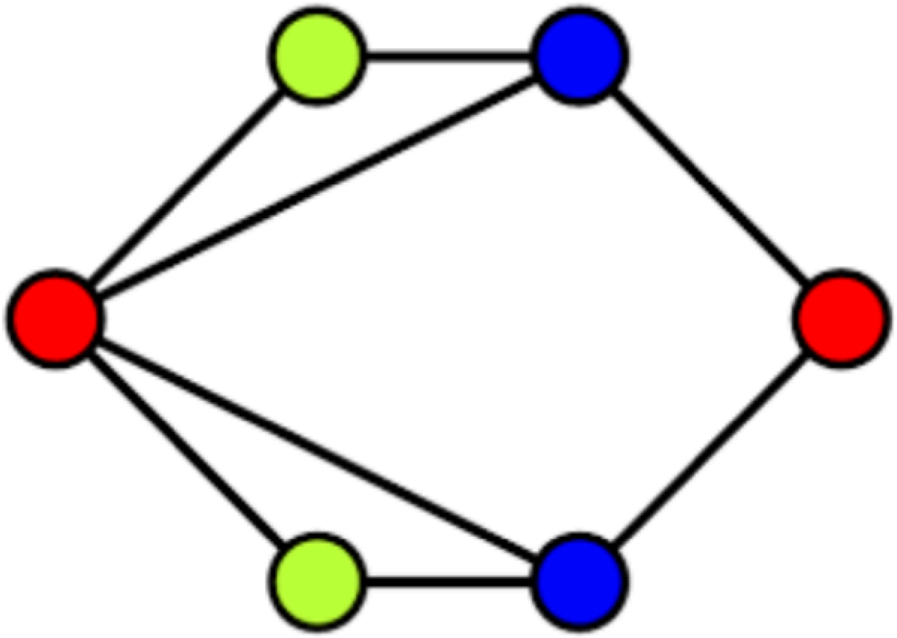
Sec: A4_B4_52

Batch: B4

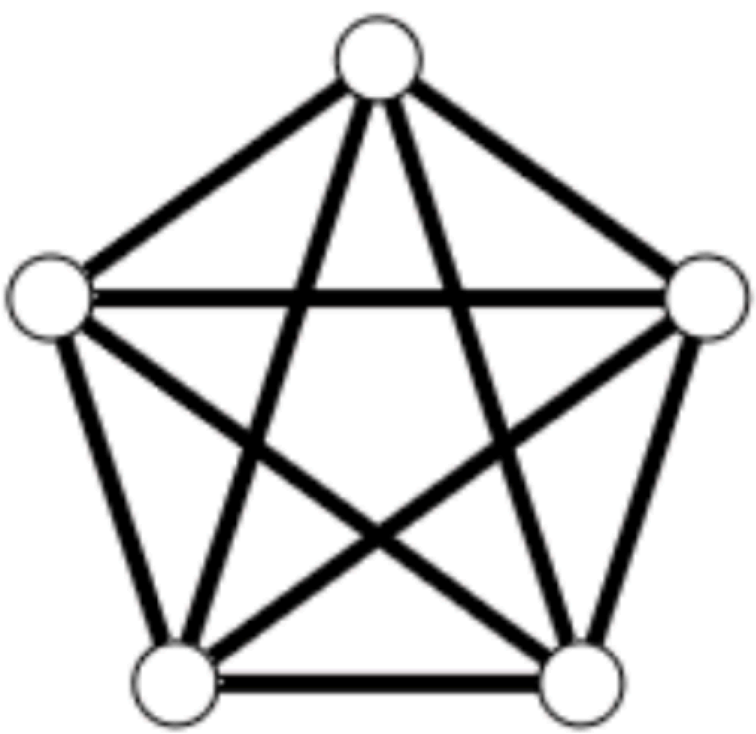
PRACTICAL NO. 8

Aim: Implement the Graph Colouring algorithm using the Graph Colouring concept.
Problem Statement:
A GSM is a cellular network with its entire geographical range divided into hexagonal cells. Each cell has a communication tower that connects with mobile phones within the cell. Assume this GSM network operates in different frequency ranges. Allot frequencies to each cell such that no adjacent cells have the same frequency range.
Consider an undirected graph G = (V, E) shown in fig. Find the colour assigned to each node
using the 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:

```
def print_solution(colors):
   print("Solution Exists: Frequencies (colors) assigned are:")
   for i in range(len(colors)):
       print(f"  Cell {i} -> Frequency {colors[i]}")


def is_safe(v, graph, colors, c):
   V = len(graph)
   for i in range(V):
       if graph[v][i] == 1 and colors[i] == c:
           return False
   return True


def solve_coloring_recursive(graph, m, colors, v):
   V = len(graph)
   if v == V:
       return True


   for c in range(1, m + 1):
       if is_safe(v, graph, colors, c):
```

```python
            colors[v] = c

            if solve_coloring_recursive(graph, m, colors, v + 1):

                return True

            colors[v] = 0



    return False



def find_coloring_solution(graph, m, graph_name):

    print(f"### {graph_name} Solution ###")

    V = len(graph)

    colors = [0] * V



    if solve_coloring_recursive(graph, m, colors, 0) == False:

        print(f"No solution exists with {m} frequencies (colors).")

    else:

        print_solution(colors)



# --- Graph 1 (from image) ---
```

```
graph_1_matrix = [

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

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

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

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

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

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

]

m_1 = 3

find_coloring_solution(graph_1_matrix, m_1, "Graph 1")



print("\n------------------------------\n")



# --- Graph 2 (K5 - Complete Graph) ---

graph_2_matrix = [

    [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]

]




m_2_fail = 4

find_coloring_solution(graph_2_matrix, m_2_fail, "Graph 2 (Attempt 1: 4 Colors)")




print("")

m_2_success = 5

find_coloring_solution(graph_2_matrix, m_2_success, "Graph 2 (Attempt 2: 5 Colors)")
```

Output:

```
### Graph 1 Solution ###
Solution Exists: Frequencies (colors) assigned are:
  Cell 0 -> Frequency 1
  Cell 1 -> Frequency 2
  Cell 2 -> Frequency 2
  Cell 3 -> Frequency 3
  Cell 4 -> Frequency 3
  Cell 5 -> Frequency 1

------------------------------

### Graph 2 (Attempt 1: 4 Colors) Solution ###
No solution exists with 4 frequencies (colors).

### Graph 2 (Attempt 2: 5 Colors) Solution ###
Solution Exists: Frequencies (colors) assigned are:
  Cell 0 -> Frequency 1
  Cell 1 -> Frequency 2
  Cell 2 -> Frequency 3
  Cell 3 -> Frequency 4
  Cell 4 -> Frequency 5
```