EXPERIMENT NO.- 7

NAME- ATHARVA PALIWAL
ROLL NO.- 40

Problem Statement: A GSM is a cellular network (shown in fig) 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 only four (1-4) different frequency ranges. Allot frequencies to each cell such that no adjacent cells have same frequency range.

Program:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
class Edge
{
      int source, dest;
      public Edge(int source, int dest) {
            this.source = source;
            this.dest = dest;
      }
}
class Graph
{
      List<List<Integer>> adjList = null;
      Graph(List<Edge> edges, int N)
      {
            adjList = new ArrayList<>();
            for (int i = 0; i < N; i++) {
                  adjList.add(new ArrayList<>());
            }
            for (Edge edge: edges)
            {
                  int src = edge.source;
                  int dest = edge.dest;
                  adjList.get(src).add(dest);
                  adjList.get(dest).add(src);
            }
      }
}
```

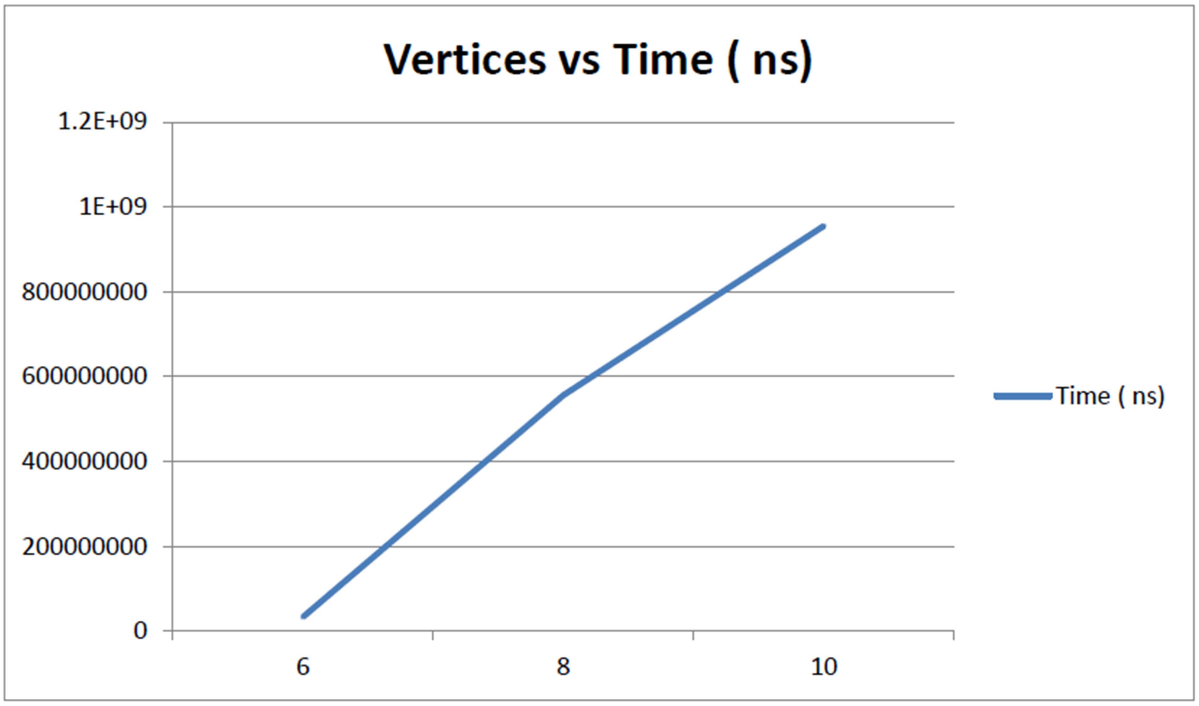
```
public class Main
{
      private static String COLORS[] = {"1","2","3","4"};
      private static boolean isSafe(Graph graph, int[] frequency, int v, int
c)
      {
            for (int u : graph.adjList.get(v))
            if (frequency[u] == c)
                  return false;
            return true;
      }
      public static void printsolution(Graph g, int[] frequency, int k, int
v,
int N)
      {
      if (v == N)
      {
            for (v = 0; v < N; v++)
                  System.out.printf("%-8s" , COLORS[frequency[v]]);
            System.out.println();
            return;
      }
      for (int c = 1; c <= k; c++)
      {
            if (isSafe(g, frequency, v, c))
            {
                  frequency[v] = c;
                  printsolution(g, frequency, k, v + 1, N);
                  frequency[v] = 0;
            }
      }
}
public static void main(String[] args)
{
      List<Edge> edges = Arrays.asList(
            new Edge(0, 1), new Edge(0, 4),
            new Edge(0, 5), new Edge(4, 5),
            new Edge(1, 4), new Edge(1, 3),
            new Edge(2, 3), new Edge(2, 4)
      );
      final int N = 6;
      Graph g = new Graph(edges, N);
      int k = 3;
      int[] frequency = new int[N];
      long timeStart=System.nanoTime();
      printsolution(g, frequency, k, 0, N);
      long timeEnd=System.nanoTime();
```

```
        System.out.println("Time Taken (in nanoseconds): "+(timeEndtimeStart));
        }
}
```

OUTPUT-

Input (No. of vertices)	No. of colors(freqrange)	Solutions	Time (nanoseconds)
6	4	1 2 1 3 3 2 1 2 1 3 3 4 1 2 1 3 4 2 1 2 1 3 4 3 1 2 1 4 3 2 1 2 1 4 3 4 1 2 1 4 4 2 1 2 1 4 4 3 1 2 2 1 3 2 1 2 2 1 3 4 1 2 2 1 4 2 1 2 2 1 4 3 1 2 2 3 3 2 1 2 2 3 3 4 1 2 2 3 4 2 1 2 2 3 4 3 1 2 2 4 3 2 1 2 2 4 3 4 1 2 2 4 4 2 1 2 2 4 4 3 1 2 3 1 4 2 1 2 3 1 4 3 1 2 3 4 4 2 1 2 3 4 4 3 1 2 4 1 3 2 1 2 4 1 3 4 1 2 4 3 3 2 1 2 4 3 3 4 1 3 1 2 2 3 1 3 1 2 2 4	34818700
8	4	1 2 1 1 3 2 3 4 1 2 1 1 3 2 4 4 1 2 1 1 3 4 3 4 1 2 1 1 3 4 4 4 1 2 1 1 4 2 3 3 1 2 1 1 4 2 4 3 1 2 1 1 4 3 3 3 1 2 1 1 4 3 4 3 1 2 2 1 3 2 3 4 1 2 2 1 3 2 4 4 1 2 2 1 3 4 3 4 1 2 2 1 3 4 4 4	567160278

		1 2 2 1 4 2 3 3 1 2 2 1 4 2 4 3 1 2 2 1 4 3 3 3 1 2 2 1 4 3 4 3 1 2 2 3 1 2 3 4 1 2 2 3 1 2 4 4 1 2 2 3 1 4 3 4 1 2 2 3 1 4 4 4 1 2 2 4 1 2 3 3 1 2 2 4 1 2 4 3 1 2 2 4 1 3 3 3 1 2 2 4 1 3 4 3 1 2 3 1 4 2 4 3 1 2 3 1 4 3 4 3 1 2 3 3 1 2 4 4 1 2 3 3 1 4 4 4 1 2 3 4 1 2 4 3 1 2 3 4 1 3 4 3	
10	4	1 2 1 3 3 2 1 2 2 3 1 2 1 3 3 2 1 2 2 4 1 2 1 3 3 2 1 2 3 2 1 2 1 3 3 2 1 2 3 4 1 2 1 3 3 2 1 2 4 2 1 2 1 3 3 2 1 2 4 3 1 2 1 3 3 2 1 3 2 3 1 2 1 3 3 2 1 3 2 4 1 2 1 3 3 2 1 3 3 2 1 2 1 3 3 2 1 3 3 4 1 2 1 3 3 2 1 3 4 2 1 2 1 3 3 2 1 3 4 3 1 2 1 3 3 2 1 4 2 3 1 2 1 3 3 2 1 4 2 4 1 2 1 3 3 2 1 4 3 2 1 2 1 3 3 2 1 4 3 4 1 2 1 3 3 2 1 4 4 2 1 2 1 3 3 2 1 4 4 3 1 2 1 3 3 2 2 2 2 3 1 2 1 3 3 2 2 2 2 4 1 2 1 3 3 2 2 2 3 2 1 2 1 3 3 2 2 2 3 4 1 2 1 3 3 2 2 2 4 2 1 2 1 3 3 2 2 2 4 3 1 2 1 3 3 2 2 3 2 3 1 2 1 3 3 2 2 3 2 4 1 2 1 3 3 2 2 3 3 2 1 2 1 3 3 2 2 3 3 4 1 2 1 3 3 2 2 3 4 2 1 2 1 3 3 2 2 3 4 3 1 2 1 3 3 2 2 4 2 3 1 2 1 3 3 2 2 4 2 4 1 2 1 3 3 2 2 4 3 2 1 2 1 3 3 2 2 4 3 4	975941000



Analysis:
The idea is to allot frequencies one by one to different vertices, starting from the vertex 0. Before assigning a frequency, check for safety by considering already assigned frequency to the adjacent vertices i.e. check if the adjacent vertices have the same frequency or not. If there is any frequency assignment that does not violate the conditions, mark the frequency assignment as part of the solution. If no assignment of frequency is possible then backtrack and return false.
Complexity Analysis:

- Time Complexity: O(m^V).
  There are total O(m^V) combination of frequency. So time complexity is O(m^V). The upper bound time complexity remains the same but the average time taken will be less.
- Space Complexity: O(V).
  To store the output array O(V) space is required.