19BCE248

Compiler Construction

Practical 10

AIM: To implement Code Optimization techniques: Implement any code optimization technique.

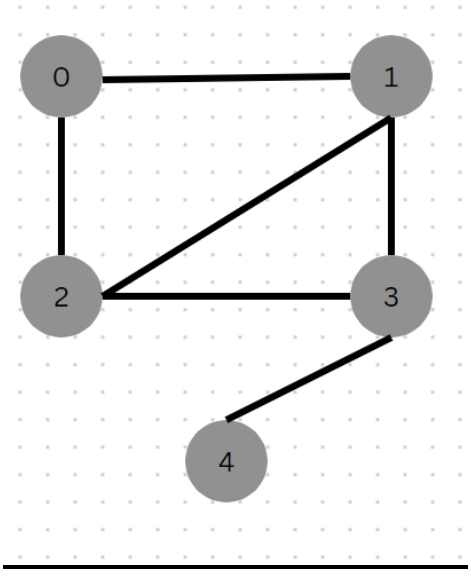
Source Code:

```
import java.util.LinkedList;
import java.util.*;
import java.io.*;


public class Graphs {
    private int N;
    private LinkedList < Integer > adjList[];


    Graphs(int n) {
        N = n;
        adjList = new LinkedList[n];
        for (int i = 0; i < n; i++) {
            adjList[i] = new LinkedList();
        }
    }

    void addingEdge(int x, int y) {
        adjList[x].add(y);
        adjList[y].add(x);
    }

    void findChromticNo(int arr[]) {
        int size = arr.length;
        Set < Integer > hashSet = new HashSet < Integer > ();
        for (int j = 0; j < size; j++) {
            hashSet.add(arr[j]);
        }
        int chromaticNo = hashSet.size();

        System.out.println("The chromatic number of the graph is: " +
chromaticNo);

    }
    void greedyColorNodes() {
        int res[] = new int[N];
```

```
        Arrays.fill(res, -1);

        res[0] = 0;
        boolean avail[] = new boolean[N];

        Arrays.fill(avail, true);

        for (int n = 1; n < N; n++) {
            Iterator < Integer > itr = adjList[n].iterator();
            while (itr.hasNext()) {
                int i = itr.next();
                if (res[i] != -1)
                    avail[res[i]] = false;
            }

            int clr;
            for (clr = 0; clr < N; clr++) {
                if (avail[clr]) {
                    break;
                }
            }

            res[n] = clr;


            Arrays.fill(avail, true);
        }


        for (int n = 0; n < N; n++) {
            System.out.println("Node " + n + " ---> Color - " + res[n]);
        }

        findChromticNo(res);
    }

    public static void main(String argvs[]) {
        Graphs graph1 = new Graphs(5);


        graph1.addingEdge(0, 1);
        graph1.addingEdge(0, 2);
        graph1.addingEdge(1, 2);
        graph1.addingEdge(1, 3);
        graph1.addingEdge(2, 3);
        graph1.addingEdge(3, 4);

        System.out.println("Coloring of the graph 1 is: ");
```

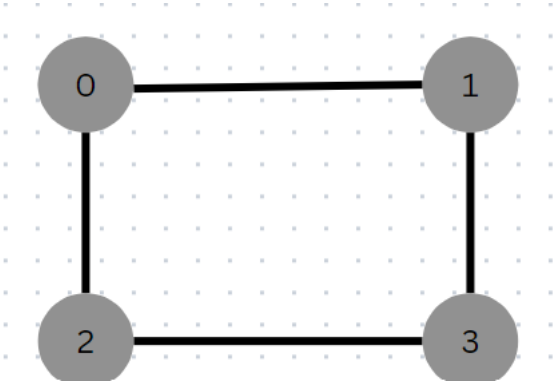
```
        graph1.greedyColorNodes();

        System.out.println();
        Graphs graph2 = new Graphs(4);

        System.out.println("Coloring of the graph 2 is: ");


        graph2.addingEdge(0, 1);
        graph2.addingEdge(0, 2);
        graph2.addingEdge(1, 3);
        graph2.addingEdge(2, 3);


        graph2.greedyColorNodes();


    }

}
```

Output:



```
Minimum Register of the graph 1 is:
Node 0 ---> Register - 0
Node 1 ---> Register - 1
Node 2 ---> Register - 2
Node 3 ---> Register - 0
Node 4 ---> Register - 1
The Minimum Register of the graph is: 3
```

Similarly for graph 2:




```
Minimum Register of the graph 2 is:
Node 0 ---> Register - 0
Node 1 ---> Register - 1
Node 2 ---> Register - 1
Node 3 ---> Register - 0
The Minimum Register of the graph is: 2
```