

Experiment 2.2

Studentname: Rohan Jaiswal

Branch: BE-CSE

Semester: 6th

Subject Name: Advanced Programming lab-2

UID: 21BCS2856

Section/Group: KRG_CC_1-B

Date of Performance: 26-02-2024

Subject Code: 21CSP-351

Aim:

1. To Solve the Gray Code Problem
2. To Solve the Is Graph Bipartite Problem

Objective:

- There is an undirected graph with n nodes, where each node is numbered between 0 and $n - 1$. You are given a 2D array `graph`, where `graph[u]` is an array of nodes that node u is adjacent to. More formally, for each v in `graph[u]`, there is an undirected edge between node u and node v . The graph has the following properties:.
- An n -bit gray code sequence is a sequence of $2n$ integers where: Every integer is in the inclusive range $[0, 2n - 1]$,

Code(A):

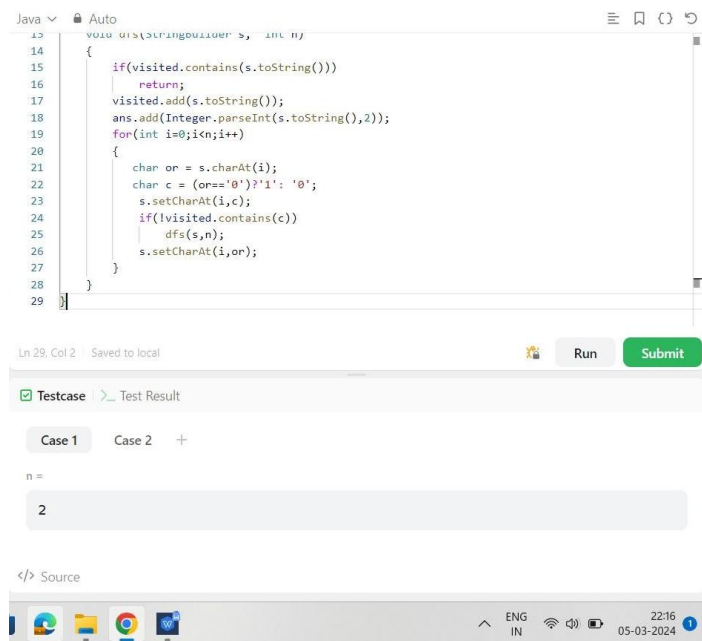
```
class Solution
{
    List<Integer> ans;
    HashSet<String> visited;
    public List<Integer> grayCode(int n)
    {
        visited = new HashSet();
        ans = new ArrayList();
        StringBuilder s = new StringBuilder();
        for(int i=0; i<n; i++)
            s.append('0');
        dfs(s, n);
        return ans;
    }
    void dfs(StringBuilder s, int n)
    {
        if(visited.contains(s.toString()))
            return;
        visited.add(s.toString());
        ans.add(Integer.parseInt(s.toString(), 2));
        for(int i=0; i<n; i++)
        {
            char or = s.charAt(i);
            char c = (or=='0')?'1': '0';
            s.setCharAt(i, c); if(!visited.contains(s.toString()))
                dfs(s, n);
            s.setCharAt(i, or);
        }
    }
}
```

```

        dfs(s,n);
        s.setCharAt(i,or);
    }
}
}

```

Output(A):



```

Java ~ Auto
13 void dfs(StringBuilder s, int n)
14 {
15     if(visited.contains(s.toString()))
16         return;
17     visited.add(s.toString());
18     ans.add(Integer.parseInt(s.toString(),2));
19     for(int i=0;i<n;i++)
20     {
21         char or = s.charAt(i);
22         char c = (or=='0')?'1':'0';
23         s.setCharAt(i,c);
24         if(!visited.contains(s.toString()))
25             dfs(s,n);
26         s.setCharAt(i,or);
27     }
28 }
29

```

Ln 29, Col 2 Saved to local Run Submit

Testcase Test Result

Case 1 Case 2 +

n =

2

</> Source

ENG IN 22:16 05-03-2024

Code(B):

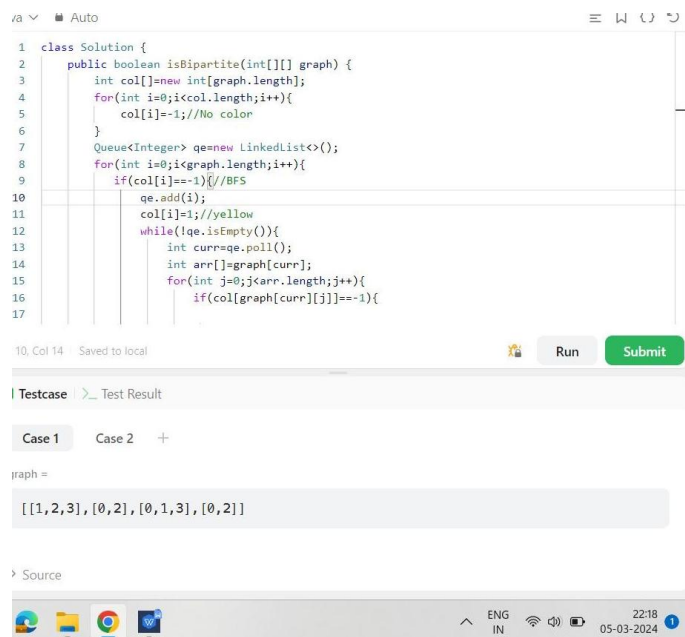
```

class Solution {
    public boolean isBipartite(int[][] graph)
    { int col[]=new int[graph.length];
      for(int i=0;i<col.length;i++){ col[i]=-1;//No color
    }
    Queue<Integer> qe=new LinkedList<>();
    for(int i=0;i<graph.length;i++){
        if(col[i]==-1){//BFS
            qe.add(i);
            col[i]=1;//yellow
            while(!qe.isEmpty()){
                int curr=qe.poll();
                int arr[]=graph[curr];
                for(int j=0;j<arr.length;j++){ if(col[graph[curr][j]]==-1){
                    col[graph[curr][j]]=1-col[curr];
                    qe.add(graph[curr][j]);
                }
            }
        }
    }
}

```

```
    }  
    else  
        if(col[graph[curr][j]]==col[curr]){ r  
            return false;  
        }  
    }  
}  
}  
}  
}  
}  
return true;  
}  
}
```

Output(B):



```
1 class Solution {  
2     public boolean isBipartite(int[][] graph) {  
3         int col[]=new int[graph.length];  
4         for(int i=0;i<col.length;i++){  
5             col[i]=-1;//No color  
6         }  
7         Queue<Integer> qe=new LinkedList<>();  
8         for(int i=0;i<graph.length;i++){  
9             if(col[i]==-1){//BFS  
10                qe.add(i);  
11                col[i]=1;//yellow  
12                while(!qe.isEmpty()){  
13                    int curr=qe.poll();  
14                    int arr[]=graph[curr];  
15                    for(int j=0;j<arr.length;j++){  
16                        if(col[graph[curr][j]]==-1){  
17
```

10, Col 14 Saved to local

Run Submit

Testcase Test Result

Case 1 Case 2 +

graph =

```
[[1,2,3], [0,2], [0,1,3], [0,2]]
```

Source

ENG IN 22:18 05-03-2024