```
1 class KosarajuSCT
 2 {
 3
       public int kosaraju(int V, ArrayList<ArrayList<Integer>> adj) {
           //code here
 4
 5
           boolean visited[]=new boolean[V]; //
           ArrayList<Integer> list = new ArrayList<Integer>();
 6
 7
 8
           // perform a dfs to get the list for decreasing order of finishing time(same list as
   a topological sort)
 9
           for(int i=0;i<V;i++){</pre>
               if(!visited[i]){
10
11
                    dfs(i,adj,visited,list);
12
               }
13
           }
14
15
           //Now reverse the graph.
           ArrayList<ArrayList<Integer>> graph = reverseGraph(V,adj);
16
17
18
           // new traversal
19
           Arrays.fill(visited, false);
20
           int connectedComponents=0;
21
           for(int i=V-1;i>=0;i--){
22
               if(!visited[list.get(i)]){
23
                    dfs2(list.get(i),graph,visited);
24
                    connectedComponents++; // every time you perform dfs it a new SCT
25
               }
26
27
            return connectedComponents;
28
29
       }
30
31
       public static void dfs(int vertex,ArrayList<ArrayList<Integer>> adj, boolean
   []visited,ArrayList<Integer> list){
32
           visited[vertex]=true;
33
           ArrayList<Integer> cl = adj.get(vertex);
34
           int len = cl.size();
35
           for(int i=0;i<len;i++){</pre>
36
               int nextNode = cl.get(i);
37
               if(!visited[nextNode]){
                    dfs(nextNode,adj, visited, list);
38
39
40
41
           list.add(vertex);
42
       public static void dfs2(int vertex,ArrayList<ArrayList<Integer>> adj, boolean []visited)
43
44
           visited[vertex]=true;
45
           ArrayList<Integer> cl = adj.get(vertex);
           int len = cl.size();
46
           for(int i=0;i<len;i++){</pre>
47
               int nextNode = cl.get(i);
48
49
               if(!visited[nextNode]){
50
                    dfs2(nextNode,adj, visited);
51
               }
52
           }
53
       }
54
55
56
       public static ArrayList<ArrayList<Integer>> reverseGraph(int
   V,ArrayList<ArrayList<Integer>> adj){
57
           ArrayList<ArrayList<Integer>> graph = new ArrayList<ArrayList<Integer>>();
```

localhost:51644 1/2

```
4/26/2021
                                                  h:\DEV\coding\practice\test.java
 58
             for(int i=0;i<V;i++){</pre>
 59
                  graph.add(new ArrayList<Integer>());
 60
             for(int i=0;i<V;i++){</pre>
 61
                  ArrayList<Integer> cl = adj.get(i);
 62
 63
                  int len = cl.size();
                  for(int j=0;j<len;j++){</pre>
 64
 65
                      addEdge(cl.get(j),i,graph);
                  }
 66
 67
             }
             return graph;
 68
 69
         }
 70
 71
         public static void addEdge(int from,int to, ArrayList<ArrayList<Integer>> graph){
 72
             graph.get(from).add(to);
 73
         }
```

74 }

localhost:51644 2/2