```java
class KosarajuSCT
{
    public int kosaraju(int V, ArrayList<ArrayList<Integer>> adj) {
        //code here
        boolean visited[]=new boolean[V]; //
        ArrayList<Integer> list = new ArrayList<Integer>();

        // perform a dfs to get the list for decreasing order of finishing time(same list as a topological sort)
        for(int i=0;i<V;i++){
            if(!visited[i]){
                dfs(i,adj,visited,list);
            }
        }

        //Now reverse the graph.
        ArrayList<ArrayList<Integer>> graph = reverseGraph(V,adj);

        // new traversal
        Arrays.fill(visited,false);
        int connectedComponents=0;
        for(int i=V-1;i>=0;i--){
            if(!visited[list.get(i)]){
                dfs2(list.get(i),graph,visited);
                connectedComponents++; // every time you perform dfs it a new SCT
            }
        }
         return connectedComponents;

    }

    public static void dfs(int vertex,ArrayList<ArrayList<Integer>> adj, boolean []visited,ArrayList<Integer> list){
        visited[vertex]=true;
        ArrayList<Integer> cl = adj.get(vertex);
        int len = cl.size();
        for(int i=0;i<len;i++){
            int nextNode = cl.get(i);
            if(!visited[nextNode]){
                dfs(nextNode,adj, visited, list);
            }
        }
        list.add(vertex);
    }
    public static void dfs2(int vertex,ArrayList<ArrayList<Integer>> adj, boolean []visited)
{
        visited[vertex]=true;
        ArrayList<Integer> cl = adj.get(vertex);
        int len = cl.size();
        for(int i=0;i<len;i++){
            int nextNode = cl.get(i);
            if(!visited[nextNode]){
                dfs2(nextNode,adj, visited);
            }
        }
    }


    public static ArrayList<ArrayList<Integer>> reverseGraph(int V,ArrayList<ArrayList<Integer>> adj){
        ArrayList<ArrayList<Integer>> graph = new ArrayList<ArrayList<Integer>>();
```

```java
        for(int i=0;i<V;i++){
            graph.add(new ArrayList<Integer>());
        }
        for(int i=0;i<V;i++){
            ArrayList<Integer> cl = adj.get(i);
            int len = cl.size();
            for(int j=0;j<len;j++){
                addEdge(cl.get(j),i,graph);
            }
        }
        return graph;
    }

    public static void addEdge(int from,int to, ArrayList<ArrayList<Integer>> graph){
        graph.get(from).add(to);
    }
}
```