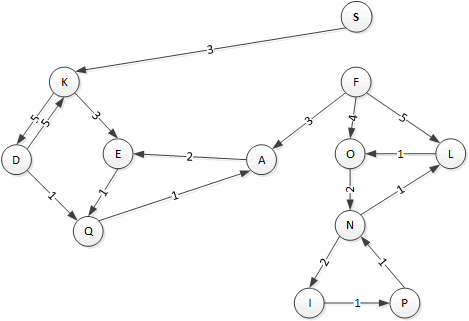
**Kosaraju's Algorithm**

**Medium**Accuracy: 49.73% Submissions: 32420 Points: 4

Given a Directed Graph with**V**vertices **(**Numbered from**0 to V-1)** and **E** edges, Find the number of strongly connected components in the graph.



**Example 1:**

**Input:**

Diagram

Description automatically generated

**Output:**

3

**Explanation**:

Diagram

Description automatically generated

We can clearly see that there are 3 Strongly

Connected Components in the Graph

**Example 2:**

**Input:**

Diagram

Description automatically generated

**Output:**

1

**Explanation**:

All of the nodes are connected to each other.

So, there's only one SCC.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **kosaraju()** which takes the number of vertices V and adjacency list of the graph as inputs and returns an integer denoting the number of strongly connected components in the given graph.

**Expected Time Complexity:** O(V+E).  
**Expected Auxiliary Space:** O(V).

**Constraints:**  
1 ≤ V ≤ 5000  
0 ≤ E ≤ (V\*(V-1))  
0 ≤ u, v ≤ N-1  
Sum of E over all testcases will not exceed 25\*106

class Solution {

    public:

    //Function to find number of strongly connected components in the graph.

    void dfs(int i, vector<vector<int>> &graph, vector<bool> &visited) {

        visited[i]=true;

        for (auto x : graph[i]) {

            if (!visited[x]) dfs(x, graph, visited);

        }

    }

    void dfs1(int i, vector<int> adj[], vector<bool> &visited, vector<int> &order) {

        visited[i]=true;

        for (auto x : adj[i]) {

            if (!visited[x]) dfs1(x, adj, visited, order);

        }

        order.push\_back(i);

    }

    int kosaraju(int V, vector<int> adj[]) {

        //code here

        vector<vector<int>> graph(V);

        for (int i=0; i<V; i++) {

            for (auto x : adj[i]) graph[x].push\_back(i);

        }

        vector<int> order;

        vector<bool> visited(V, false);

        for (int i=0; i<V; i++) {

            if (!visited[i]) dfs1(i, adj, visited, order);

        }

        int count=0;

        for (int i=0; i<V; i++) visited[i]=false;

        for (int i=V-1; i>=0; i--) {

            if (!visited[order[i]]) {

                count++;

                dfs(order[i], graph, visited);

            }

        }

        return count;

    }

};