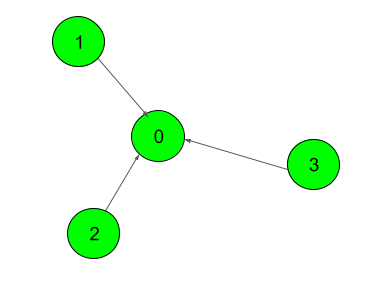
**Topological sort** :-

Medium Accuracy: 56.52% Submissions: 144K+ Points: 4

Given a Directed Acyclic Graph (DAG) with V vertices and E edges, Find any Topological Sorting of that Graph.

**Example 1:**

**Input:**



**Output:**

1

**Explanation**:

The output 1 denotes that the order is

valid. So, if you have, implemented

your function correctly, then output

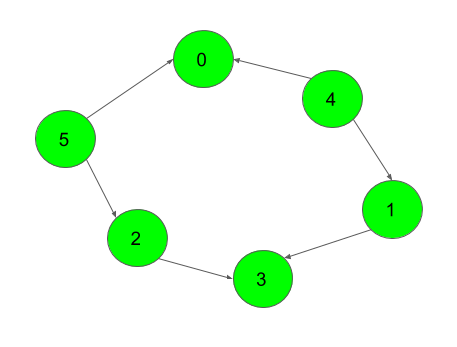
would be 1 for all test cases.

One possible Topological order for the

graph is 3, 2, 1, 0.

**Example 2:**

**Input:**



**Output:**

1

**Explanation:**

The output 1 denotes that the order is

valid. So, if you have, implemented

your function correctly, then output

would be 1 for all test cases.

One possible Topological order for the

graph is 5, 4, 2, 1, 3, 0.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **topoSort()**  which takes the integer V denoting the number of vertices and adjacency list as input parameters and returns an array consisting of the vertices in Topological order. As there are multiple Topological orders possible, you may return any of them. If your returned topo sort is correct then the console output will be 1 else 0.

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

**Constraints:**  
2 ≤ V ≤ 104  
1 ≤ E ≤ (N\*(N-1))/2

**Code** :-

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution

{

public:

unordered\_map<int, vector<int>> mp;

unordered\_map<int, bool> mila;

vector<int> s;

//vector<int> ans;

void func(int t){

//base-case no parent found

if(mp[t].empty()==true){

if(mila[t]==false){

s.push\_back(t);

mila[t]=true;

}

return;

}

//recursive-program

while(mp[t].empty()==false){

int p = mp[t].back();

mp[t].pop\_back();

func(p);

}

if(mila[t]==false){

s.push\_back(t);

mila[t]=true;

}

return;

}

//Function to return list containing vertices in Topological order.

vector<int> topoSort(int V, vector<int> adj[])

{

for(auto i=0; i<V; i++){

if(adj[i].size()!=0){

for(auto ind:adj[i]){

mp[ind].push\_back(i);

}

}

}

for(auto i=0; i<V; i++){

func(i);

}

return s;

}

};

//{ Driver Code Starts.

/\* Function to check if elements returned by user

\* contains the elements in topological sorted form

\* V: number of vertices

\* \*res: array containing elements in topological sorted form

\* adj[]: graph input

\*/

int check(int V, vector <int> &res, vector<int> adj[]) {

if(V!=res.size())

return 0;

vector<int> map(V, -1);

for (int i = 0; i < V; i++) {

map[res[i]] = i;

}

for (int i = 0; i < V; i++) {

for (int v : adj[i]) {

if (map[i] > map[v]) return 0;

}

}

return 1;

}

int main() {

int T;

cin >> T;

while (T--) {

int N, E;

cin >> E >> N;

int u, v;

vector<int> adj[N];

for (int i = 0; i < E; i++) {

cin >> u >> v;

adj[u].push\_back(v);

}

Solution obj;

vector <int> res = obj.topoSort(N, adj);

cout << check(N, res, adj) << endl;

}

return 0;

}

// } Driver Code Ends