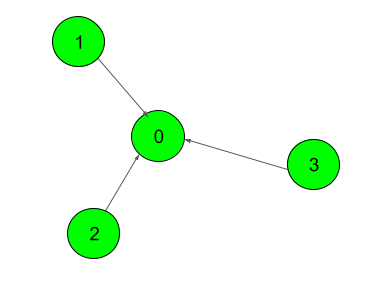
**Topological sort**

**Medium**

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

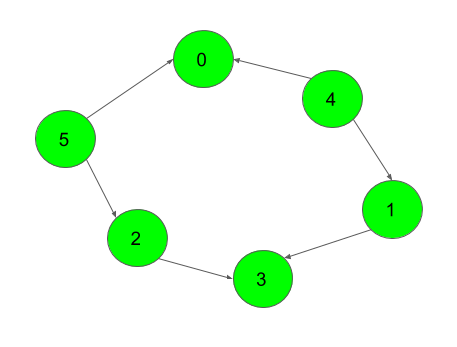
would be 1 for all test cases.

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.

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

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

**Company Tags**

[**Moonfrog Labs**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Moonfrog%20Labs) [**Flipkart**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Flipkart) [**Morgan Stanley**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Morgan%20Stanley) [**Accolite**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Accolite) [**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft) [**OYO Rooms**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=OYO%20Rooms) [**Samsung**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Samsung) [**D-E-Shaw**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=D-E-Shaw) [**Visa**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Visa)

//{ Driver Code Starts

import java.util.\*;

import java.io.\*;

import java.lang.\*;

class Main {

public static void main(String[] args) throws IOException {

BufferedReader read =

new BufferedReader(new InputStreamReader(System.in));

int t = Integer.parseInt(read.readLine());

while (t-- > 0) {

ArrayList<ArrayList<Integer>> list = new ArrayList<>();

String st[] = read.readLine().trim().split("\\s+");

int edg = Integer.parseInt(st[0]);

int nov = Integer.parseInt(st[1]);

for (int i = 0; i < nov; i++)

list.add(i, new ArrayList<Integer>());

int p = 0;

for (int i = 1; i <= edg; i++) {

String s[] = read.readLine().trim().split("\\s+");

int u = Integer.parseInt(s[0]);

int v = Integer.parseInt(s[1]);

list.get(u).add(v);

}

int[] res = new Solution().topoSort(nov, list);

if (check(list, nov, res) == true)

System.out.println("1");

else

System.out.println("0");

}

}

static boolean check(ArrayList<ArrayList<Integer>> list, int V, int[] res) {

if(V!=res.length)

return false;

int[] map = new int[V];

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

map[res[i]] = i;

}

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

for (int v : list.get(i)) {

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

}

}

return true;

}

}

// } Driver Code Ends

/\*Complete the function below\*/

class Solution

{

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

static int[] topoSort(int V, ArrayList<ArrayList<Integer>> adj)

{

int[] ans=new int[V];

int[] indeg=new int[V];

for(ArrayList<Integer>list :adj){

for(Integer node : list){

indeg[node]++;

}

}

Queue<Integer> q=new LinkedList<>();

int k=0;

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

if(indeg[i]==0){

q.add(i);

}

}

while(!q.isEmpty()){

int curr=q.poll();

ans[k++]=curr;

for(int neighours : adj.get(curr)){

if(--indeg[neighours]==0){

q.add(neighours);

}

}

}

return ans;

}

}