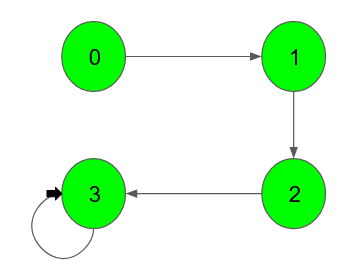
**Detect cycle in a directed graph**

**Medium**

Given a Directed Graph with **V** vertices (Numbered from **0** to **V-1**) and **E** edges, check whether it contains any cycle or not.

**Example 1:**

**Input:**

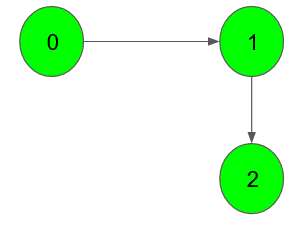


**Output:** 1

**Explanation**: 3 -> 3 is a cycle

**Example 2:**

**Input:**



**Output:** 0

**Explanation**: no cycle in the graph

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

**Constraints:**  
1 ≤ V, E ≤ 105

**Company Tags**

[**Flipkart**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Flipkart) [**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft) [**Samsung**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Samsung) [**MakeMyTrip**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=MakeMyTrip) [**Oracle**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Oracle) [**Goldman Sachs**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Goldman%20Sachs) [**Adobe**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Adobe) [**BankBazaar**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=BankBazaar) [**Rockstand**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Rockstand)

//{ Driver Code Starts

import java.util.\*;

import java.io.\*;

import java.lang.\*;

class DriverClass {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int t = sc.nextInt();

while (t-- > 0) {

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

int V = sc.nextInt();

int E = sc.nextInt();

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

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

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

int u = sc.nextInt();

int v = sc.nextInt();

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

}

if (new Solution().isCyclic(V, list) == true)

System.out.println("1");

else

System.out.println("0");

}

}

}

// } Driver Code Ends

/\*Complete the function below\*/

class Solution {

public boolean isCyclic(int V, ArrayList<ArrayList<Integer>> adj) {

boolean[] visited=new boolean[V];

boolean[] dfsvisited=new boolean[V];

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

if(!visited[i] && helperCyclic(i, visited, dfsvisited, adj)){

return true;

}

}

return false;

}

private boolean helperCyclic(int u, boolean[] visited, boolean[] dfsvisited, ArrayList<ArrayList<Integer>> adj){

visited[u]=true;

dfsvisited[u]=true;

for(int list :adj.get(u)){

if(!visited[list] && helperCyclic(list, visited, dfsvisited, adj)){

return true;

}

else if(dfsvisited[list]){

return true;

}

}

dfsvisited[u]=false;

return false;

}

}