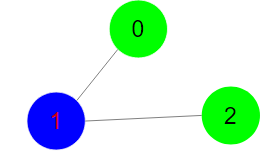
**Bipartite Graph**

**Medium**

Given an adjacency list of a graph**adj**of V no. of vertices having 0 based index. Check whether the graph is bipartite or not.

**Example 1:**

**Input:**



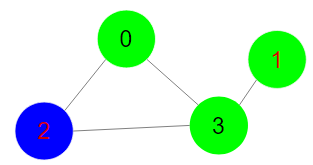
**Output:** 1

**Explanation:** The given graph can be colored

in two colors so, it is a bipartite graph.

**Example 2:**

**Input:**



**Output:** 0

**Explanation:** The given graph cannot be colored

in two colors such that color of adjacent

vertices differs.

**Expected Time Complexity:**O(V + E)  
**Expected Space Complexity:**O(V)  
  
**Constraints:**  
1 ≤ V, E ≤ 105

**Company Tags**

[**Flipkart**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Flipkart) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft) [**Samsung**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Samsung)

//{ Driver Code Starts

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class CodingMaxima

{

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

{

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

int T = Integer.parseInt(br.readLine().trim());

while(T-->0)

{

String[] S = br.readLine().trim().split(" ");

int V = Integer.parseInt(S[0]);

int E = Integer.parseInt(S[1]);

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

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

adj.add(new ArrayList<Integer>());

}

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

String[] s = br.readLine().trim().split(" ");

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

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

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

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

}

Solution obj = new Solution();

boolean ans = obj.isBipartite(V, adj);

if(ans)

System.out.println("1");

else System.out.println("0");

}

}

}

// } Driver Code Ends

class Solution

{

boolean flag=false;

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

{

int[] color=new int[V];

Arrays.fill(color, -1);

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

if(color[i]==-1){

color[i]=0;

flag=true;

if(!helper(i, adj, color))

return false;

}

}

return true;

}

public boolean helper( int srs ,ArrayList<ArrayList<Integer>>adj, int[] color ){

int coloridx=color[srs];

for(int neborious: adj.get(srs)){

if(color[neborious]!=-1){

if(color[neborious]==coloridx){

flag=false;

return false;}

}

else{

color[neborious]=color[srs]==0?1:0;

helper(neborious, adj, color);

}

}

return flag;

}

}