CSE 208
January 2025

Practice on BFS and DFS

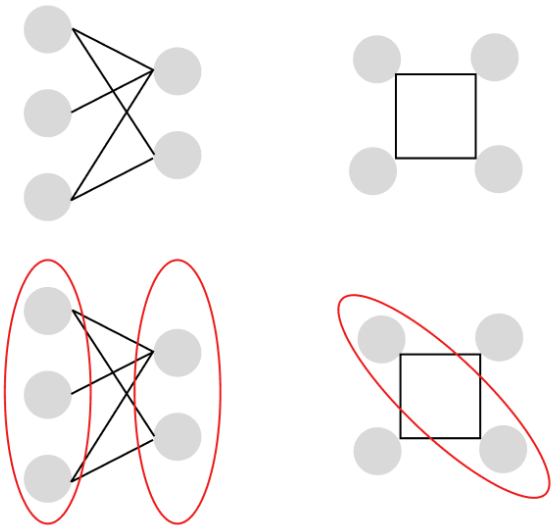
Problem 1:

Given an undirected graph, determine if it is bipartite or not.

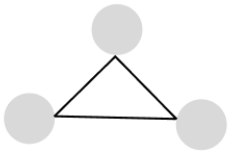
Bipartite:

A graph is called bipartite if you can divide its vertices into two disjoint sets U and V, such that every edge in the graph connects a vertex in set U to a vertex in set V. No two vertices within the same set are adjacent. Moreover, a bipartite graph does not contain any odd-length cycles.

Consider the following examples:



In both cases, we can divide the vertices into two sets without violating the properties of a bipartite graph. Notice that the graphs do not contain any odd cycles.



The above graph contains an odd-length cycle. We cannot divide the vertices into two sets such that there are no edges between the vertices of the same set.

Input format

The first line contains two integers, v and e where ‘v’ denotes the number of vertices, and ‘e’ denotes the number of edges. Each of the following ‘m’ lines represents an edge.

Output format

Print “Bipartite” (without the quotes) if the graph is bipartite otherwise print “Not Bipartite”.

Sample I/O:

Input	Output
4 4 0 2 0 3 1 2 1 3	Bipartite
3 3 0 1 1 2 2 0	Not Bipartite
6 7 0 1 0 4 1 2 1 5 2 3 3 5 4 5	Bipartite
6 7 0 1 0 3 1 2 1 5 2 4 3 4 3 5	Not Bipartite

Problem 2:

Two brilliant hackers, Elliot and Darlene, are trying to hack into the impenetrable network of Fort Knox, to grab some highly confidential information. They have brainstormed some potential security loopholes (i.e. vulnerabilities) and established a cause and effect relationship among them. Specifically, they have modeled the potential loopholes and the relation among them as a directed graph whose nodes denote the potential vulnerabilities and the edges denote the dependencies of one vulnerability on another. Now, as they are being predictive, they have come up with thousands of such graphs. To proceed further, they need to analyze some information obtained by traversing each of these graphs. To facilitate the process, they have designed an automated graph traversal agent, which they call ToMBOT. The ToMBOT is extremely efficient in traversing some graphs, but there is a catch. Given an acyclic graph (i.e. tree), it can operate as expected but gets stuck otherwise. So, Elliot and Darlene have thought of passing the ToMBOT only those graphs that can be processed by it and dealing with the remaining ones differently. As they are occupied with finding more possible vulnerabilities, they have asked for your help. Now, can you help them by determining whether a graph can be passed to the ToMBOT or not?

Input format

The first line contains 2 integers, n and m where n denotes the number of potential security loopholes and m denotes the number of dependencies between two loopholes. Each of the next m lines contains two integers u, v (0 <= u, v < n) which indicates, vulnerability v is a direct consequence of vulnerability u.

Output format

Print “Yes” (without the quotes) if the input can be processed by the ToMBOT and “No” otherwise.

Sample I/O

Input	Output
4 5 0 1 1 2 0 2 1 3 3 2	Yes
3 3 0 1 1 2 2 0	No
3 2 0 1 2 1	Yes