1. Graph Building

Given a directed graph, print its adjacency information. You can use an array of vectors in c++ to store the graph in a adjacency list. Below are a few tips to use it.

Declaration:

```
vector<int> g[100];
```

(Assuming your nodes are numbered 0 to 99. You need to change the size, if then number of nodes are different)

Initialization:

```
for(i=0;i<N;i++)g[i].clear();
```

(Assuming you have N nodes in total numbered 0 to N-1. g[i].clear() function empties the i-th vector)

Adding edge:
To add a particular edge u -> v:

```
g[u].push_back(v);
```

Traversing the list of edges for a particular node u:

```
for(i=0;i<g[u].size();i++){
	v = g[u][i]; // so you have an edge from u to v
}
```

Input
First line: N (0<N<=100000), number of nodes.
Second line: M (0<M<-100000), number of edges.
Next M lines, each: U V (0.U, V<N), defines an edge from U to V.

Output
Total N lines, each for one node. Print the adjacent nodes in the input order for that node. See sample for more clarification.

Sample

Input	Output
6 6 1 2 1 3 0 1 2 4 4 5 5 1	0: 1 1: 2 3 2: 4 3: 4: 5 5: 1

2. Shortest Path Length

Given an unweighted directed graph, find shortest path from a single source node.

Input
First line: N (0<N<=100000), number of nodes.
Second line: M (O<M<=200000), number of edges.
Next M lines, each: U V (0<=U, V<N), defines an edge from U to V.

Output
Total N lines, each for one node. Print the shortest path of each node considering node 0 as the source. See sample for more clarification.

Sample

Input	Output
6 6 1 2 1 3 0 1 2 4 5 4 5 1	0: 0 1: 1 2: 2 3: 2 4: 3 5: Not Reachable

3. Components, Discovery and Fininshing Time

Given an UNWEIGHTED UNDIRECTED graph, run dfs and find the components with discovery time and finishing time. Assume that you are running dfs in increasing order.

Input First line:
N (0<N<=1000), number of nodes.
Second line: M (O<M<=20000), number of edges.
Next M lines, each: U V (0<=U, V<N), defines an edge from U to V.

Output
Total N lines, each for one node. Print the component number, discovery time and finishing time each node. See sample for more clarification.

Sample

Input	Output
6 6 1 2 1 3 0 4 2 3 5 4 5 0	0: 1 1 6 1: 2 7 12 2: 2 8 11 3: 2 9 10 4: 1 2 5 5: 1 3 4

4. Cycle Detection

Given an undirected graph, find if the graph has any cycle.

Input
First line: N (0<N<=100000), number of nodes.
Second line: M (O<M<=200000), number of edges.
Next M lines, each: U V (0<=U, V<N), defines an edge from U to V.

Output
Print Yes/No. See sample for more clarification.

Samples

Input 6 4 1 2 1 3 0 4 5 4	Output No
Input 3 3 1 2 2 0 0 1	Output Yes