A

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 (0<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.

|  |  |
| --- | --- |
| 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 |

B

Given an unweighted undirected graph, run DFS and find the component no for each node. Assume that you are running DFS in increasing order. Nodes which are connected via path belongs to the same component.

**Input**

First line: N (0<N<=1000), number of nodes.

Second line: M (0<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 for each node. See sample for more clarification.

|  |  |
| --- | --- |
| 6  6  1 2  1 3  0 4  2 3  5 4  5 0 | 0: 1  1: 2  2: 2  3: 2  4: 1  5: 1 |

**Explanation:**

In the sample, Node 0, 4, 5 belongs to the first component. Node 1, 2, 3 belongs to second component.

C

Given an unweighted undirected graph, find if the graph is bipartite. In a bipartite is graph, the vertices can be divided into two sets A and B, such that, there will be no edges between two nodes of set A. Also there will be no edges between two nodes of set B.

**Input**

First line: N (0<N<=100), number of nodes.

Second line: M (0<M<=2000), number of edges.

Next M lines, each: U V (0<=U, V<N), defines an edge from U to V.

**Output**

Yes/No. See sample for clarification

|  |  |
| --- | --- |
| 6  5  1 3  1 2  0 1  3 4  2 4 | Yes |
| 3  3  0 1  0 2  1 2 | No |

Explanation:

In the first sample, the sets are {1, 4} and {0, 2, 3}.

D

Given a directed graph, classify its edges**. You should run DFS in increasing order**. The graph will not contain multiple edges between a pair of nodes.

**Input**

First line: N (0<N<=100), number of nodes.

Second line: M (0<N<=4950), number of edges.

Next M lines, each: U V (0<=U, V<N), defines an edge from U to V.

**Output**

Edge classifications. Print edges in increasing order. Here, T means tree edge, B means back edge, F means forward edge and C means cross edge. See sample for clarification.

|  |  |
| --- | --- |
| 6  7  0 1  1 2  2 0  2 3  0 3  4 5  5 1 | 0 1 (T)  0 3 (F)  1 2 (T)  2 0 (B)  2 3 (T)  4 5 (T)  5 1 (C) |

**Hint:**

To run DFS in increasing order, either you need to use adjacency matrix or you need to sort each adjacency vector. Given the limits, using adjacency matrix will be easier for this problem.

E

Given an NxM grid, with blocked cells, find the largest connected empty area. For each empty cell, it is connected with four adjacent cells (if empty): Up, Down, Left and Right. You can never step into a blocked cell.

**Input**

First line: N (0<N<=100), M (0<M<=100).

Next N lines, each contains M characters. ‘.’ indicates empty cell. ‘#’ indicates blocked cells.

**Output**

One line with the largest connected area. See sample for more clarification.

|  |  |
| --- | --- |
| 5 3  ...  ##.  ...  .##  ... | 11 |
| 5 3  ...  ###  ...  .#.  ... | 8 |

F

Given an unweighted undirected unrooted tree, find its diameter. The diameter of a tree is the distance between the nodes which are furthest apart.

**Input**

First line: N (0<N<=100,000), number of nodes.

Next N-1 lines, each: U V (0<=U, V<N), defines an edge between U and V.

**Output**

Diameter. See sample for clarification

|  |  |
| --- | --- |
| 6  1 3  1 2  0 1  3 4  2 5 | 4 |

**Explanation:**

The most distant nodes are 4 and 5 and the distance between them is 4.