CSN-261: Data Structures Laboratory

Lab Assignment 11 (L11)

---

1. For an undirected graph G having E edges and V vertices, write a Java program to check whether G is *2-edge connected* or not. A graph is called *2-edge connected* if it remains connected on removing any edge.

   Input

   - Number of vertices, V.
   - Number of edges, E.
   - Adjacency matrix of the graph G.

   Output

   'Yes' if the graph G is 2-edge connected, 'No' otherwise.

   Sample Input

   4
   6

   $$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

   Sample Output

   'Yes'

2. Implement a code in java using the following information. Given a connected and undirected graph. A spanning tree of a given graph is a subgraph that is a tree that connects all the vertices without any cycle. There may be many different spanning trees of a single graph, but a minimum spanning tree (MST) or with minimum weight spanning tree for a weighted, connected, and an undirected graph is a spanning tree with weight less than or equal to the weight of every other spanning tree.

Input

- Line 1 contains T, representing the number of test cases.
- First line of each test case has two space-separated integers V and E indicates the Number of vertices and the Number of edges, respectively.
- Next K lines have three space-separated integers X, Y, W, which indicates an edge between the vertex X and Y of weight W.
- Vertices will be labeled starting from 1. For example, if $V = 5$, the set of vertices is $\{1, 2, 3, 4, 5\}$.

Output

- Output should consist of V-1 lines where each line would represent an edge that is part of the MST. Each of the output lines should have three space-separated integers X, Y, and W representing an edge between X and Y, which is weight W.
- The edges should be printed in increasing weights. Also, print such that $X < Y$ for each edge.

Constraints

- $1 \leq T \leq$100
- $2 \leq V \leq$1000
- $1 \leq E \leq$V*(V-1)/2
- $1 \leq X, Y \leq$V
- $1 \leq W \leq 10^6$

Sample Input

```
2
4 5
1 2 10
1 3 6
3 4 4
2 4 15
1 4 5
5 7
```

```
1 2 24
2 3 9
3 4 8
4 5 28
1 5 10
1 4 25
2 5 30
```

Sample Output

```
3 4 4
1 4 5
1 2 10
3 4 8
2 3 9
1 5 10
1 2 24
```

3. There are two friends, say A and B. Assume that friend A lives in the house u i.e., u^{th} vertex and B lives in the house n i.e., n^{th} vertex. For group study, friend A has to visit the house of friend B daily. After a few days, friend A notices that there are few edges such that he passes them every time he goes to meet friend B, no matter which path he takes. You have been given information about the map in the form of houses network, i.e., undirected graph vertices. For each i^{th} query, you are provided u (A's house vertex), and you have to find out how many edges should be visited in every path from u to n (i.e. last vertex). If there is no such edge, print 'impossible.' Implement the code in Java. **Note:** u can be equal to n.

   Input

   - The first line contains N and M, the number of vertices and edges, respectively.
   - Next M lines contain two space-separated integers u and v, denoting there is an edge between these two vertices.
   - Next line contains Q. Next Q lines contain u.

   Output

- For each query print the answer in a separate line.

Constraints

- $1 \leq N \leq$100000
- $1 \leq M \leq$200000
- $1 \leq u \leq$N
- $1 \leq Q \leq$100000
  Note: Given graph is connected with no self-loops, multiple edges and cycles.

Sample Input

```
4 3
1 2
1 3
2 4
4
1
2
3
4
```

Sample Output

```
2
1
3
impossible
```