582 Randomly Wired Neural Nets

Introduction

The following story ("AI Koan") is from *The Hacker's Dictionary:*

> In the days when Sussman was a novice, Minsky once came to him as he sat hacking at the PDP-6.
>
> "What are you doing?", asked Minsky.
>
> "I am training a randomly wired neural net to play Tic-Tac-Toe" Sussman replied.
>
> "Why is the net wired randomly?", asked Minsky.
>
> "I do not want it to have any preconceptions of how to play", Sussman said.
>
> Minsky then shut his eyes.
>
> "Why do you close your eyes?", Sussman asked his teacher.
>
> "So that the room will be empty."
>
> At that moment, Sussman was enlightened.

A Graph Model

Graphs are powerful tools for modeling connected entities of all kinds (including randomly wired neural nets). In this problem, we wish to prune unnecessary nodes from a graph that is modeling a particular kind of neural net. In particular, we wish to remove neurons from the neural net that do not have any potential across them (or, equivalently, any flow through them) when we introduce a potential across two given nodes of the network.

Let $G = (V, E)$ be an undirected graph with $|V| = n$ vertices and $|E| = m$ edges. For two given distinct vertices v and w of G, a path $P(v, w)$ in G between v and w is called a *simple path* between v and w if no vertex of G appears in $P(v, w)$ more than once.

It is straightforward to show that a neuron in a neural net will support flow if and only if the corresponding edge in the graph model of the neural net lies on some simple path between v and w. Thus, the set of all vertices of G that are on at least one simple path between v and w represent the vertices that would be present in a model of the neural net with the unnecessary neurons removed.

Your program is to print out a modified graph G such that each vertex in the graph is present on at least one simple path between v and w.

Input

The input will consist of a graph G described as an adjacency list and two distinguished vertices, v and w. The first line of the input will contain two numbers separated by a space, indicating the distinguished vertices, v and w. The second line will contain one number n, specifying the number of vertices in G. The next n lines specify the adjacency list for the graph, one line for each vertex, ordered from vertex 0 to vertex $n - 1$. The line corresponding to vertex q (i.e., the qth line in the adjacency list portion of the input file) consists of an integer k specifying the degree of q (the number of edges connected to q), followed by k integers specifying the vertices connected to vertex q. All of the numbers in one line will be separated by a single space.

The maximum number of vertices is 1024.

Output

The output of the program will consist of an ordered list of all vertices of G such that each of the identified vertices is on some simple path in G between v and w.Lines 0 through $n-1$ should specify the connectivity for the vertices 0 through $n-1$, respectively. If a given vertex is not present in the modified graph, an `X` should be printed on that line. Otherwise, the vertices (in the modified graph) to which the given vertex is connected should be printed.

Sample Input

```
0 6
8
2 1 4
5 0 2 3 5 7
1 1
1 1
2 0 5
3 4 1 7
1 7
3 1 5 6
```

Sample Output

```
1 4
0 5 7
X
X
0 5
4 1 7
7
1 5 6
```