Maximum in Each Vertex

Problem Code	hw02c_nodemax
Running Time Limit	1 sec
Memory Limit	16 mb

Objective

- Be able to read graph data structure in adjacency list format
- Be able to iterate over graph structure

Introduction

This problem is the first problem directly involving graph. A weighted directed graph structure is given and the task is to identify all edges with maximum weight.

A graph consists of vertices and edges. Each edge connects two vertices. In a directed graph, edge has a direction such that an edge connecting from vertex A to vertex B is not the same as an edge connecting from B to A. In weighted graph, a weighted is associated with each edge. This identifies that that edge has specific "weight".

In programming problem, it is convenient to label each vertex in a graph with N vertices by number $0, 1, \dots, N - 1$.

Task

Your task is to identify all edges with maximum weight.

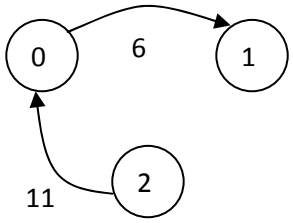
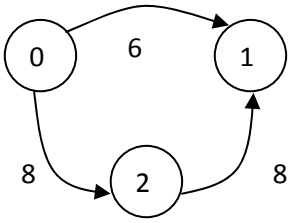
Input

The input is the graph itself. The first line of input indicates the number of vertices $(1 \leq N \leq 100)$. There will be N more lines. The $(i + 1)^{th}$ line describes the edges going out of vertex i. The first number M_i $(0 \leq M_i < N)$ in the line gives the number of edges from vertex i. There will be M_i pairs of numbers in the line. Each pair d_{ij} w_{ij} describes each edge by giving the destination vertex and the associated weight. It is guaranteed that $0 \leq d_{ij} < N$ and w_{ij} can be stored in `int` variable.

The generalized format of the input can be expressed as follow.

N
M_0 d_{00} w_{00} d_{01} w_{01} … d_{0M_0-1} w_{0M_0-1}
M_1 d_{10} w_{10} d_{11} w_{11} … d_{1M_1-1} w_{1M_1-1}
…
M_{n-1} d_{n-10} w_{n-10} d_{n-11} w_{n-11} … $d_{n-1M_{n-1}-1}$ $w_{n-1M_{n-1}-1}$

The following table demonstrates graphs and their represented input.

	
3 1 1 6 0 1 0 11	3 2 1 6 2 8 0 1 1 8

Output

The first line of output gives the number of edges with maximum weight. There must be that many lines after the first line. Each line gives two numbers which is the starting vertex and the destination vertex of the edge. The edge should be sorted by the starting vertex first, then by destination vertex. For example, if we wish to display 3 edges (2,1), (1,7) and (1,5). The output should be

```
3
1 5
1 7
2 1
```

Example

Ex1

Input	Output
3 1 1 6 0 1 0 11	1 2 0

Ex2

Input	Output
3 2 1 6 2 8 0 1 1 8	2 0 2 2 1