Problem 6 - Recover Graph (100 pts)

Time Limit	:	1 s
Memory Limit	:	1024 MB

Problem Description

Robert, a clever student, is learning graph theory in the DSA course. Commonly, people use adjacency lists to store the graph. Robert is diligent, so he tries to implement adjacent lists after class. Formally, Robert implements it with the pseudo code in the following:

Algorithm 1: Implementation of adjacency lists

```
Function BuildAdjList(numVertices, edgeList):
    adjList[1..numVertices] ← empty lists
    for edge (x, y) in edgeList do
        insert y into adjList[x]
        insert x into adjList[y]
    end
    return adjList[1.. numVertices]
```

However, after getting the adjacency list, Robert forgets the original list. He would like to verify the correctness of his implementation. Help Robert to recover the original list of edges, in the original given order.

Input

The first line contains only one integer N ($1 \leq N \leq 10^5$), representing the number of vertices in the graph. The vertices are indexed from 1 to N.

Each of the next N lines contains the description of one of the adjacency lists. The first integer num_i in the i^{th} line represents the length of the list of vertex i. Then, the following num_i numbers represents the neighbors of vertex i in the list. You can assume that $0 \leq \sum num_i \leq 4 \cdot 10^5$. It is guaranteed that without considering the order of vertices in the adjacency lists, the graph is simple, that is, no self-loops and multiple edges exist.

Output

You should determine if there exist any possible list of edges satisfying the constraints given in the input. If no answers exists, print "No" to the output (without quotation marks).
Otherwise, print "Yes" in the first line (without quotation marks). In this case, in the following lines, print two integers on the i^{th} line, representing the i^{th} edge in the list of edges. If there are more then one list of edges satisfying the constraints, you may print any of them.

Subtask 1 (30 pts)

- $N \leq 1000$.

Subtask 2 (70 pts)

- No other constraints.

Sample Input 1

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

Sample Output 1

```
Yes
1 2
2 3
1 3
```

Sample Input 2

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

Sample Output 2

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

Sample Input 3

```
6
2 5 2
3 1 5 3
4 4 5 6 2
3 3 5 6
5 4 2 6 1 3
3 4 3 5
```

Sample Output 3

```
No
```