AC Automation Chicken

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	1024 megabytes

Braised Chicken loves Aho-Corasick automatons. Therefore, he came up with the following problem related to Aho-Corasick automatons. Before we dive into the problem, he kindly reminds you of the following definitions:

- A **Trie** T is a rooted tree, on each edge of which a character is written. A vertex x on the Trie should not have two children y and z such that the characters written on the edges (x, y) and (x, z) are the same.
- Suppose there is a given Trie T rooted at r. For a node x, **the string represented by** x is the resultant string when you concatenate the characters on the edges that are on the path in T from r to x, in the order they appear on the path. Particularly, the string represented by r is the empty string. It can be proved that no two different vertices represent equal strings.
- We say a string S **exists** in a Trie T if and only if there exists a vertex x in T such that the string represented by x is S.
- A **fail tree** F of a given Trie T is a rooted tree whose root is the root of T, r. Define S_x as the string represented by node x. For a non-root node x, suppose U is the longest proper suffix of S_x (a proper suffix of a string S is a suffix of S that is not equal to S) that exists in T. Then, $fail_x$ is defined as the vertex of T such that $S_{fail_x} = U$. Note that the empty suffix of S_x always exists in T, so $fail_x$ always exists. The edge set of F is $\{(x, fail_x) \mid x \in [1, n], x \neq r\}$. It can be proven that these edges form a tree.

Braised Chicken has a Trie T of n vertices, numbered 1 through n. You do not know its root. The character set is all the integers in $[1, n]$. Then, he built the corresponding fail tree F of the Trie. After that, he combined the edges of T (directed away from the root) and the edges of F (directed towards the root) to get a **unweighted directed graph** G with n nodes and $2n - 2$ directed edges. To be more specific, G is constructed as follows:

- For every non-root vertex u, G contains an edge $fa_u \to u$ where fa_u is u's father on T.
- For every non-root vertex u, G contains an edge $u \to fail_u$.

And finally comes your task: given the unweighted directed graph G of n vertices (the edges are given in an arbitrary order), you need to

- find out the root of T, vertex r;
- identify which edges of G are on T and which edges are on F;
- find a valid way of writing a character (an integer in $[1, n]$) on each edge of T so that T and its corresponding fail tree indeed accord with G.

If there is no valid way of assigning the root and information of the edges, you should also report this fact.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 5 \times 10^4$). Description of the test cases follows.

The first line of each test case contains an integer n ($1 \leq n \leq 10^5$).

Then $2n-2$ lines follow. Each of these $2n-2$ lines contains two integers x, y ($1 \leq x, y \leq n$), denoting an edge in G that goes from x to y.

The sum of n over all test cases does not exceed 5×10^5.

Output

For each test case, if there is no valid way of assigning the root and information of the edges, print `No`.

Otherwise, print `Yes` in the first line. Then print $n-1$ lines. Each of these $n-1$ lines should contain three integers x, y, z ($1 \leq x, y, z \leq n$), denoting that T has an edge from x to y (x should be the father of y) with the character z written on it. The vertex without any incoming edges is the root of T. It should hold that the edges you output form a rooted Trie corresponding with G.

You can print the edges in any order. If there are multiple ways of assigning T, print any.

Example

standard input	standard output
4 4 1 4 4 1 1 2 2 1 2 3 3 4 2 1 2 1 2 1 3 1 2 2 1 2 3 3 2	Yes 1 2 1 2 3 2 4 1 1 No Yes Yes 1 2 1 2 3 1

Note

In the first test case, the root of T is 4. It can be verified that $fail_1 = 4, fail_2 = 1, fail_3 = 4$.

Note that there are also other valid outputs for this input: for example, letting 1 be the root and T have edges $(1 \to 2, 1), (2 \to 3, 2), (1 \to 4, 2)$ (here, $(x \to y, z)$ means that the father of y is x, and character z is written on the edge from x to y) is also a valid answer.