Little Garden

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

The No Names, facing a desperate situation, have decided to summon Sakamaki Izayoi and two others to Little Garden to aid in their plight. Upon arrival, the first thing that Izayoi notices is the wide variety of trees. Sensing their interest, Laplace's Demon decides to send a message containing some snippets of a book about the trees of Little Garden. Izayoi doesn't bother, but his companions Asuka and Kasukabe decide to engage. Below is their discussion. (**X1** denotes Asuka, and **X2** denotes Kasukabe. If you are intimidated by the length of the statement, focus on Kasukabe's part, as Asuka's part is primarily to provide clarity.)

X1: Let me read out his message.

> Definition: The tree described in this book are defined inductively: a single node constitutes a tree, a tree can be used as the left (or right) child of a parent to constitute a tree, and two trees can be used as the left and right children of a parent respectively to form a tree. All structures generated only by the above rules with limited steps are called trees.

X2: In other words, in this book, "trees" refer to binary trees that **are not empty, have a root, and distinguish between left and right children**.

X1: Exactly. The next section discusses the isomorphism of two trees.

> Definition: Call two trees T, T' isomorphic, denoted by $T \equiv T'$, if one of the following four rules apply.
>
> 1. Trees composed of individual nodes are isomorphic with each other;
> 2. If the root nodes of two trees have only a left subtree, and their left subtrees are isomorphic, then the two trees are isomorphic;
> 3. If the root nodes of two trees have only a right subtree, and their right subtrees are isomorphic, then the two trees are isomorphic;
> 4. If the root nodes of two trees have left and right subtrees, and their left and right subtrees correspond to isomorphisms, then the two trees are isomorphic.
>
> For convenience, we treat isomorphic trees as the same tree.

X2: This means that the nodes of the trees are unlabelled. If two trees have the same structure (*including in differentiating between left and right children*), they are considered the same, even if the labels of the nodes are different. Otherwise they are considered different.

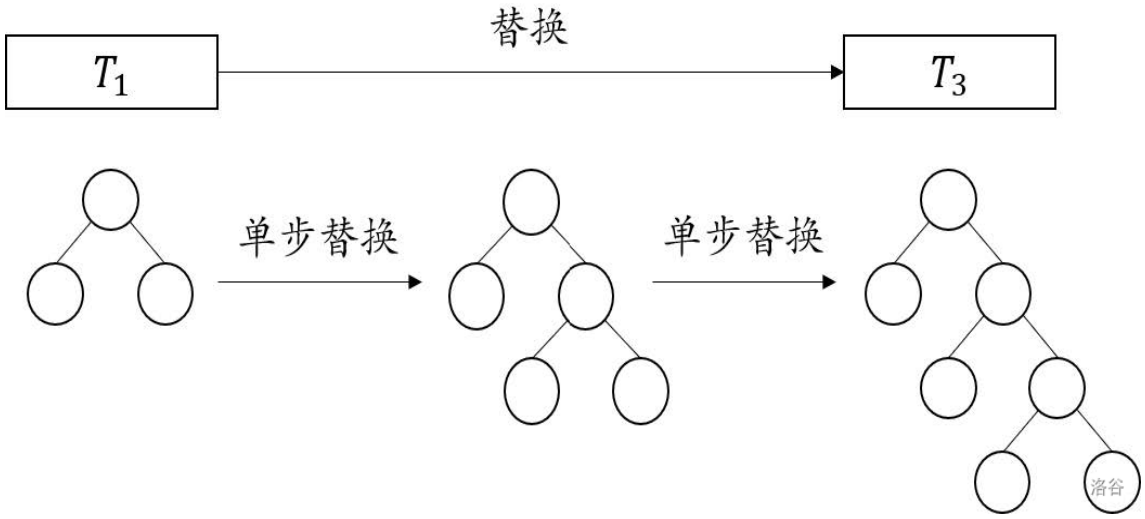
X2: I see that the leaves of a tree are also defined. Like the usual definition, a leaf refers to a node without any children. Does this really need to be defined separately?

X1: I'm not quite disgusted with this. Compared with the "intuition" based on experience, accurate definition and rigorous proof are still more reassuring. You see, the next definition is not so intuitive.

> Definition: Call it a *single-step replacement* of a tree T into T' if one leaf node of T is replaced by a tree T'' to form T'. This operation can be denoted by $T \to T'$.
>
> Call it a *replacement* of a tree T into T' if there is a natural number $n \geq 1$ and trees $T_1, T_2, \ldots, T_n$ such that $T \equiv T_1 \to T_2 \to \cdots \to T_n = T'$.

X2: In other words, a single-step replacement deletes a leaf node and places another tree in the corresponding position, as if the leaf node "grows out" a larger subtree. A tree can be replaced by another

tree if after a sequence of zero, one or more single-step replacements, we can get that tree. The following diagram should help clarify the situation.



In this diagram, we see that via two single-step replacements, we go from T_1 to T_3. Therefore, T_1 can be replaced by T_3.

X1: You are right. In particular, any tree can be replaced by an infinite number of different trees, and a tree composed of only one node can be replaced by any tree. There are also other definitions in the book.

> Definition: For a tree T, define $\text{grow}(T)$ to represent the set of trees that can be replaced by T, that is, $\text{grow}(T) = \{T' \mid T \to^\star T'\}$. One step further, if $\theta = \{T_1, T_2, \ldots, T_n\}$ is a finite set of trees, then $\text{grow}(\theta)$ is the union of all $\text{grow}(T_i)$ where $i = 1, 2, \ldots, n$, which is
>
> $$\text{grow}(\theta) = \bigcup_{T_i \in \theta} \text{grow}(T_i)$$

X2: Let's call $\text{grow}(\theta)$ the set of trees grown from the initial set of trees θ. In other words, $\text{grow}(T)$ contains all the trees that can replace T, and $\text{grow}(\theta)$ is the union of all $\text{grow}(T)$, containing all the trees that can replace any tree $T \in \theta$.

We may wish to call the collection of trees a forest. To be less rigorous, the new forest that a forest grows is the forest that all the trees in it can grow in all possible ways. Obviously, the forest that a non-empty forest grows is an infinite forest. But this infinite forest, or $\text{grow}(\theta)$ does not necessarily contain all trees. Furthermore, it does not even necessarily contain "almost all" trees.

X1: Let me add: we call a forest almost complete (or almost contains all trees), if only a limited number of trees are not in it. For a finite forest θ, $\text{grow}(\theta)$ either contains all trees, or contains almost all trees, or there are infinite trees not in it. If this is an OI question, the questioner will definitely give examples of three situations in the sample. The key theorem in the book also uses the same definition as ours.

> Theorem (Almost Complete Decidability): A set of trees is almost complete if only a finite number of trees are not among them. Then, for a given finite set of trees θ, there is an efficient algorithm to determine whether $\text{grow}(\theta)$ is almost complete.

X2: This problem has become a pure OI problem! Let me restate the meaning of the question in our language: **Given a forest of finite size θ, determine whether** $\text{grow}(\theta)$ **is almost complete, that is, whether there are only a limited number of trees that cannot be grown by the trees contained in the forest.**

X1: In other words, given a finite set of trees θ, determine whether there is only a finite number of trees T satisfying $T \notin \text{grow}(\theta)$, which means there is no $T' \in \theta$, such that $T' \to^\star T$. This is indeed very different from the typical OI problem. I cannot even think of an algorithm to solve this problem yet.

X2: It is the same for me. But I have not felt such an urge to solve a problem in quite a long time...

Input

There are multiple test cases per input file. The first line contains a single integer T, denoting the number of test cases. Next, there are T test cases, and each test case has the following format:

The first line is a positive integer m, which represents the number of trees in the test case. Next, there are m trees, each in the following format:

The first line is a positive integer n, which represents the number of nodes in the tree. The node numbers are $1, 2, \ldots, n$. The next n lines contain two non-negative space-separated integers on each line, where the i-th line contains l_i and r_i, respectively representing the numbers of the left and right child nodes of node i. If the left (or right) child does not exist, then l_i (or r_i) is 0. Of course, a leaf node must satisfy $l_i = r_i = 0$.

The input data is guaranteed to form a tree with node 1 as the root node. Please note: the labelling of nodes is only for the convenience of input, any isomorphic trees are considered the same. There may be trees that are isomorphic to each other in the input; if these duplicate trees are removed (that is, only one of each isomorphic tree is left), they can form a tree set θ. You need to determine whether the set $\text{grow}(\theta)$ generated by this tree set is almost complete.

Output

The output contains T lines, which represent the answers to the T test cases. Among them, line i outputs a string: if the set of trees grown by the set of trees input in the i-th test case is almost complete (in other words, there are only finite trees that cannot be grown by it), then output Almost Complete; otherwise, output No. Please pay attention to the spelling and capitalisation of the output string.

Scoring

$\sum m \le 2 \times 10^6$, $\sum n \le 2 \times 10^6$, $\max h \le 2 \times 10^6$, $T \le 10^2$.

$\sum n$ represents the sum of the number of nodes of all trees appearing in an input file; $\sum m$ represents the number of trees appearing in an input file; $\max h$ represents the maximum height of all the trees that appear in this input file (the height of the tree that contains only one node is 1).

- Subtask 1 [8 points]: $\max h = 1$
- Subtask 2 [16 points]: $\max h \le 2$, $m \le 4$ for every test case
- Subtask 3 [5 points]: $\max h \le 4$
- Subtask 4 [5 points]: $\max h \le 10$
- Subtask 5 [6 points]: Constraint 1, $\sum m \le 10^6$, $\sum n \le 10^6$
- Subtask 6 [18 points]: Constraint 2, $\sum m \le 2 \times 10^5$, $\sum n \le 2 \times 10^5$
- Subtask 7 [14 points]: $\sum m \le 2 \times 10^3$, $\sum n \le 2 \times 10^3$
- Subtask 8 [8 points]: $\sum m \le 6 \times 10^5$, $\sum n \le 6 \times 10^5$
- Subtask 9 [20 points]: No additional constraints.

Constraint 1: For all trees in the input, each non-leaf node has exactly one child - in other words, all the trees are chains.

Constraint 2: For all trees in the input, one of the following cases is satisfied:

- Each non-leaf node has at most one child - the tree is a chain.
- There are exactly two leaf nodes, they have the same parent node, and apart from these three nodes, the other nodes have exactly one child.

Examples

standard input	standard output
1 1 1 0 0	Almost Complete
1 3 3 2 3 0 0 0 0 2 2 0 0 0 2 0 2 0 0	Almost Complete
1 2 3 2 3 0 0 0 0 2 2 0 0 0	No

Note

In the first sample testcase, the tree consisting of a single node can be replaced by any arbitrary tree in a single step. The set grow(θ) hence consists of all trees. Thus, the answer is Almost Complete.

In the second sample testcase, it is possible to verify that we are able to construct all trees except the tree consisting of a single node. Since there is only a finite number of trees that we cannot construct, the answer is Almost Complete.

In the third sample testcase, every tree whose root has a right child but no left child cannot be constructed from the two trees given. There are an infinite number of trees of this form and thus the answer is No.