No More BSTs! (Hard Version)

Time Limit: 1 Second
Memory Limit: 2048 MB

If you have seen the normal version of this problem in this class before, the only difference between the two versions is the constraints on n and m.

You just created a binary search tree from an array of size n using the knowledge you learned from CS 495. However, LetianPie doesn't like BSTs so he wants to destroy your BST. In specific, you know that he will choose m vertices $v_1, \ldots, v_m$ (the vertices are not necessarily distinct), and for each vertex v_i, he will perform one of the following operations:

- $+\ v_i\ k$: add k to all vertices in the subtree rooted at v_i.
- $-\ v_i\ k$: subtract k from all vertices in the subtree rooted at v_i.

You don't want LetianPie to destroy your BST, so you want to know if he will reach his goal after the operations. For each operation, check if the new tree is still a valid BST after LetianPie has performed the operation (as well as all operations beforehand).

A tree is a valid BST if for all vertices v, all vertices in its left subtree have values less than or equal to the value of v, and all vertices in its right subtree have values greater than or equal to the value of v. For simplicity, the tree is rooted at vertex 1.

Input

The first line of input contains two integers n and m ($1 \leq n, m \leq 10^5$) - the number of vertices in the BST and the number of operations.

The second line contains n integers $a_1, \ldots, a_n$ ($1 \leq a_i \leq 10^5$), where a_i is the value of vertex i. It is guaranteed that a_i are distinct.

The third line contains $n - 1$ integers $f_2, \ldots, f_n$ ($1 \leq f_i \leq n$), where vertex f_i is the father of vertex i. It is guaranteed that the provided tree is a BST.

The next m lines describe LetianPie's operations. Each line begins with one of $+$ and $-$, followed by two integers v and k ($1 \leq v \leq n$, $1 \leq k \leq 10^5$), as described in the problem statement.

Output

For each operation, output `YES` if the tree is still a BST after the operation, and `NO` otherwise.

Sample Inputs

```
5 3
3 1 5 2 4
1 1 2 3
- 3 1
- 5 2
+ 3 10
```

Sample Outputs

```
YES
NO
YES
```