D. Tree Requests

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Roman planted a tree consisting of n vertices. Each vertex contains a lowercase English letter. Vertex 1 is the root of the tree, each of the $n - 1$ remaining vertices has a *parent* in the tree. Vertex is connected with its parent by an edge. The parent of vertex i is vertex p_i, the parent index is always less than the index of the vertex (i.e., $p_i < i$).

The *depth* of the vertex is the number of nodes on the path from the root to v along the edges. In particular, the depth of the root is equal to 1.

We say that vertex u is in the *subtree* of vertex v, if we can get from u to v, moving from the vertex to the parent. In particular, vertex v is in its subtree.

Roma gives you m queries, the i-th of which consists of two numbers v_i, h_i. Let's consider the vertices in the subtree v_i located at depth h_i. Determine whether you can use the letters written at these vertices to make a string that is a *palindrome*. The letters that are written in the vertexes, can be rearranged in any order to make a palindrome, but all letters should be used.

Input

The first line contains two integers n, m $(1 \le n, m \le 500\,000)$ — the number of nodes in the tree and queries, respectively.

The following line contains $n - 1$ integers $p_2, p_3, ..., p_n$ — the parents of vertices from the second to the n-th $(1 \le p_i < i)$.

The next line contains n lowercase English letters, the i-th of these letters is written on vertex i.

Next m lines describe the queries, the i-th line contains two numbers v_i, h_i $(1 \le v_i, h_i \le n)$ — the vertex and the depth that appear in the i-th query.

Output

Print m lines. In the i-th line print "`Yes`" (without the quotes), if in the i-th query you can make a palindrome from the letters written on the vertices, otherwise print "`No`" (without the quotes).

Examples

input

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

output

```
Yes
No
Yes
```

```
Yes
Yes
```

Note

String s is a *palindrome* if reads the same from left to right and from right to left. In particular, an empty string is a palindrome.

Clarification for the sample test.

In the first query there exists only a vertex 1 satisfying all the conditions, we can form a palindrome "z".

In the second query vertices 5 and 6 satisfy condititions, they contain letters "c" and "d" respectively. It is impossible to form a palindrome of them.

In the third query there exist no vertices at depth 1 and in subtree of 4. We may form an empty palindrome.

In the fourth query there exist no vertices **in subtree of 6 at depth 1**. We may form an empty palindrome.

In the fifth query there vertices 2, 3 and 4 satisfying all conditions above, they contain letters "a", "c" and "c". We may form a palindrome "cac".