F. Subtree Minimum Query

time limit per test: 6 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given a rooted tree consisting of n vertices. Each vertex has a number written on it; number a_i is written on vertex i.

Let's denote $d(i, j)$ as the distance between vertices i and j in the tree (that is, the number of edges in the shortest path from i to j). Also let's denote the *k-blocked subtree* of vertex x as the set of vertices y such that both these conditions are met:

- x is an ancestor of y (every vertex is an ancestor of itself);
- $d(x, y) \le k$.

You are given m queries to the tree. i-th query is represented by two numbers x_i and k_i, and the answer to this query is the minimum value of a_j among such vertices j such that j belongs to k_i-blocked subtree of x_i.

Write a program that would process these queries quickly!

Note that the queries are given in a modified way.

Input

The first line contains two integers n and r ($1 \le r \le n \le 100000$) — the number of vertices in the tree and the index of the root, respectively.

The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le 10^9$) — the numbers written on the vertices.

Then $n - 1$ lines follow, each containing two integers x and y ($1 \le x, y \le n$) and representing an edge between vertices x and y. It is guaranteed that these edges form a tree.

Next line contains one integer m ($1 \le m \le 10^6$) — the number of queries to process.

Then m lines follow, i-th line containing two numbers p_i and q_i, which can be used to restore i-th query ($1 \le p_i, q_i \le n$).

i-th query can be restored as follows:

Let $last$ be the answer for previous query (or 0 if $i = 1$). Then $x_i = ((p_i + last) \bmod n) + 1$, and $k_i = (q_i + last) \bmod n$.

Output

Print m integers. i-th of them has to be equal to the answer to i-th query.

Example

input

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

output

```
2
5
```