Problem
Art of Data Structures
Time limit: 5 seconds

Little Cyan Fish is teaching the Data Structure Master Class at the University of Cup. In traditional data structure problems, you would now be handed a pile of queries and asked to evaluate some convoluted expression on a fixed data structure. Oh, come on... who wants to do that in 2026? Little Cyan Fish wants to do something different. He asks you to invent the data structure on your own.

Your task is to construct a rooted binary tree $\mathcal{T}$:

- Every *internal vertex*[*] of $\mathcal{T}$ has exactly two children.
- $\mathcal{T}$ has exactly m leaves, labeled from 1 to m.

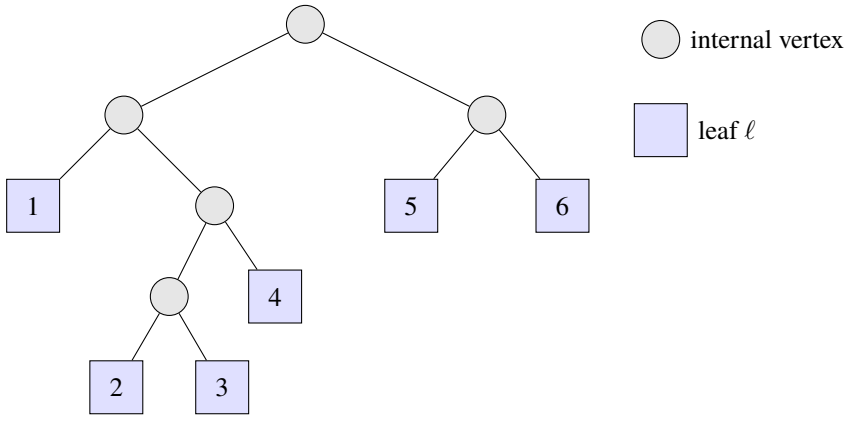


Figure 1: A valid $\mathcal{T}$ for $m = 6$. Every internal vertex has exactly two children, and the leaves carry the labels $1, \ldots, m$ in some order. Here the depth is 5.

For any set S of leaf labels, define its cost on $\mathcal{T}$ as the number of internal vertices v of $\mathcal{T}$ such that the subtree of v contains both:

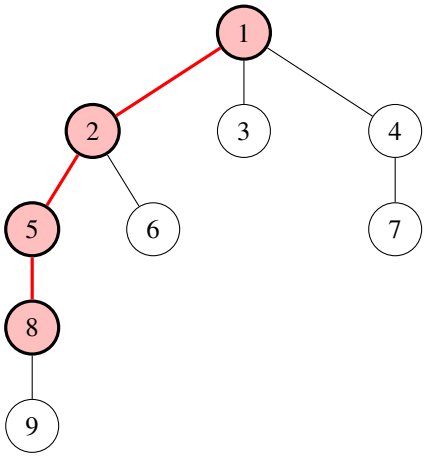
- At least one leaf whose label belongs to S.
- At least one leaf whose label does not belong to S.

Little Cyan Fish gives you two rooted trees T_1 and T_2. Both trees have vertices labeled from 1 to n, and vertex 1 is the root of each tree. He also gives you m ordered pairs (x_i, y_i), where x_i is a vertex of T_1 and y_i is a vertex of T_2. The leaf labeled ℓ in $\mathcal{T}$ is associated with the values x_ℓ and y_ℓ.

For a rooted tree T and a vertex x, let $\text{path}(T, x)$ be the set of vertices on the path from x to the root of T, including both endpoints.

Little Cyan Fish wants you to know that, for each vertex u of T_1, define $Q_1(u) = \{\ell \mid x_\ell \in \text{path}(T_1, u)\}$. Similarly, for each vertex u of T_2, define $Q_2(u) = \{\ell \mid y_\ell \in \text{path}(T_2, u)\}$. Each $Q_i(u)$ is a set of leaf labels of $\mathcal{T}$.

[*]A *leaf* is a vertex with no children, and an *internal vertex* is a vertex that is not a *leaf*.



With $x = 8$: path$(T_i, 8) = \{8, 5, 2, 1\}$ (highlighted in red).

Figure 2: The set path(T_i, x) contains every vertex on the unique path from x up to the root, including both endpoints.

The sets that Little Cyan Fish checks are $Q_1(u)$ and $Q_2(u)$ for every $1 \le u \le n$. Little Cyan Fish accepts your data structure if it satisfies both requirements:

- the depth of every vertex in $\mathcal{T}$ is at most 100, where the root has depth 1;
- among all $2n$ checked sets, the maximum cost is at most 16 666.

Show Little Cyan Fish that you are the real master of data structures!

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 10^6$).

The next line of the input contains $n - 1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$), describing the tree T_1. The integer p_i is the parent of vertex i in T_1.

The next line of the input contains $n - 1$ integers $p'_2, p'_3, \ldots, p'_n$ ($1 \le p'_i < i$), describing the tree T_2. The integer p'_i is the parent of vertex i in T_2.

The next m lines describe the ordered pairs. The i-th of these lines contains two integers x_i and y_i ($1 \le x_i, y_i \le n$).

Output

Output a single line containing a sequence of integers that describes the binary tree $\mathcal{T}$ you construct.

- A leaf labeled i is described by the integer i.
- An internal vertex is described by the integer 0, followed by the description of its left subtree, then by the description of its right subtree.

Under this encoding, every integer from 1 to m must appear exactly once, and each occurrence of 0 represents one internal vertex.

For example, the sequence 0 1 0 2 3 describes a tree whose root has leaf 1 as its left child and an internal vertex as its right child; that internal vertex has leaves 2 and 3 as its children.

Sample Input 1

```
1 1


1 1
```

Sample Output 1

```
1
```

Sample Input 2

```
3 3
1 1
1 2
1 1
2 2
3 3
```

Sample Output 2

```
0 1 0 2 3
```

Sample Input 3

```
5 8
1 2 3 4
1 1 1 1
1 1
2 1
3 2
4 2
5 3
5 5
1 5
3 4
```

Sample Output 3

```
0 0 1 0 0 3 8 0 2 7 0 4 0 5 6
```

Explanation of Sample 1: The binary tree has a single leaf labeled 1. Its depth is 1, and every possible query has cost 0.

This page is intentionally left blank.