Rank Queries on a Tree (ktree)

You are given a tree with N nodes indexed from 1 to N, rooted in node 1. Each node has a value V_i.

Giorgio asked you to answer Q queries of the form u, v, k: Which is the k-th largest value among those of the nodes on the path between nodes u and v (including endpoints)? Here, $k = 1$ corresponds to the largest value, $k = 2$ to the second largest, etc.

> ☞ Among the attachments of this task you may find a template file `ktree.*` with a sample incomplete implementation.

Input

The first line contains the integer N: the number of nodes in the tree.

The second line contains N integers V_i: the i-th integer represents the value of node i.

The third line contains $N - 1$ integers P_i: the i-th integer represents P_{i+1}, the parent of node $i + 1$.

The fourth line contains a single integer Q: the number of queries.

The following Q lines contain 3 integers u_j, v_j, k_j each, representing a query.

Output

You have to output Q lines, the j-th of them containing a single integer: the answer to the j-th query.

Constraints

- $2 \le N \le 250\,000$.
- $1 \le V_i \le N$ for each $i = 1 \dots N$.
- $1 \le P_i \le N$ for each $i = 2 \dots N$.
- $2 \le Q \le 250\,000$.
- $1 \le u_j, v_j \le N$ for each $j = 1 \dots Q$.
- $1 \le k_j \le \text{dist}(u_j, v_j) + 1$ for each $j = 1 \dots Q$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (8 points) $N, Q \le 1\,000$

– **Subtask 3** (11 points) $N, Q \le 40\,000$, $P_i = i - 1$ for each $i = 2 \dots N$

– **Subtask 4** (18 points) $N, Q \leq 40\,000$

– **Subtask 5** (13 points) $N, Q \leq 100\,000$, $P_i = i - 1$ for each $i = 2 \ldots N$

– **Subtask 6** (20 points) $N, Q \leq 100\,000$

– **Subtask 7** (12 points) $P_i = i - 1$ for each $i = 2 \ldots N$

– **Subtask 8** (18 points) No additional limitations.

Examples

input	output
6 3 6 6 6 6 6 5 1 1 1 1 5 4 2 2 2 4 2 1 1 1 4 3 2 4 3 1	6 6 3 6 6
10 5 10 1 1 2 5 9 8 2 9 8 2 7 9 5 1 1 8 5 10 10 3 2 1 4 2 6 8 1 3 4 2 2 9 2 9 7 2 3 3 1 6 4 4 5 5 1 3 8 1	9 5 8 9 8 8 1 5 2 10

Explanation

In the **first sample case**:

- In the first query, nodes $4, 1, 5, 2$ are on the path between 4 and 2, their values are, respectively, $6, 3, 6, 6$. The second largest of them is 6.
- In the second query, nodes $2, 5, 1, 4$ are on the path between 2 and 4, their values are, respectively, $6, 6, 3, 6$. The second largest of them is 6.
- In the third query, only node 1 is on the path between 1 and 1. Its value is 3, the largest value on the path is therefore 3.

- In the fourth query, nodes $4, 1, 3$ are on the path between 4 and 3, their values are, respectively, $6, 3, 6$. The second largest of them is 6.
- In the fifth query, nodes $4, 1, 3$ are on the path between 4 and 3, their values are, respectively, $6, 3, 6$. The second largest of them is 6.