G. Yet Another Maxflow Problem

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

In this problem you will have to deal with a very special network.

The network consists of two parts: part A and part B. Each part consists of n vertices; i-th vertex of part A is denoted as A_i, and i-th vertex of part B is denoted as B_i.

For each index i ($1 \le i < n$) there is a directed edge from vertex A_i to vertex A_{i+1}, and from B_i to B_{i+1}, respectively. Capacities of these edges are given in the input. Also there might be several directed edges going from part A to part B (but never from B to A).

You have to calculate the maximum flow value from A_1 to B_n in this network. Capacities of edges connecting A_i to A_{i+1} might sometimes change, and you also have to maintain the maximum flow value after these changes. Apart from that, the network is fixed (there are no changes in part B, no changes of edges going from A to B, and no edge insertions or deletions).

Take a look at the example and the notes to understand the structure of the network better.

Input

The first line contains three integer numbers n, m and q ($2 \le n, m \le 2 \cdot 10^5$, $0 \le q \le 2 \cdot 10^5$) — the number of vertices in each part, the number of edges going from A to B and the number of changes, respectively.

Then $n - 1$ lines follow, i-th line contains two integers x_i and y_i denoting that the edge from A_i to A_{i+1} has capacity x_i and the edge from B_i to B_{i+1} has capacity y_i ($1 \le x_i, y_i \le 10^9$).

Then m lines follow, describing the edges from A to B. Each line contains three integers x, y and z denoting an edge from A_x to B_y with capacity z ($1 \le x, y \le n$, $1 \le z \le 10^9$). There might be multiple edges from A_x to B_y.

And then q lines follow, describing a sequence of changes to the network. i-th line contains two integers v_i and w_i, denoting that the capacity of the edge from A_{v_i} to A_{v_i+1} is set to w_i ($1 \le v_i < n$, $1 \le w_i \le 10^9$).

Output

Firstly, print the maximum flow value in the original network. Then print q integers, i-th of them must be equal to the maximum flow value after i-th change.

Example

input

```
4 3 2
1 2
3 4
5 6
2 2 7
1 4 8
4 3 9
1 100
2 100
```

output

```
9
14
14
```

Note

This is the original network in the example: