Truck Driver

Hungary is a country with N cities numbered from 0 to $N-1$.

The cities are connected by $N-1$ *bidirectional* roads, numbered from 0 to $N-2$. Road j ($0 \le j \le N-2$) connects city $U[j]$ and city $V[j]$ and has length $T[j]$, that is, it allows one to travel between the cities in $T[j]$ units of time. Each road connects two different cities, and each pair of cities is connected by at most one road.

A **path** between two distinct cities a and b is a sequence of *distinct* cities $p_0, p_1, \ldots, p_l$ such that:

- $p_0 = a$,
- $p_l = b$,
- for each i ($0 \le i < l$), there is a road connecting cities p_i and p_{i+1}.

It is possible to travel from any city to any other city by using the roads, that is, there is a path between every two distinct cities. Note that the path connecting any pair of cities is unique.

The **distance** of cities a and b is notated by $d(a, b)$ and defined as follows:

- if $a = b$ then $d(a, b) = 0$,
- otherwise $d(a, b)$ is the total length of the roads connecting consecutive cities in the path between a and b.

Karcsi is a truck driver who has to complete some number of deliveries in the cities. For each i from 0 to $N-1$, inclusive, Karcsi has to complete $W[i]$ deliveries in city i. Karcsi starts from city 0, and he is allowed to complete the deliveries in any order, after which he returns to city 0. A **delivery plan** is a (possibly empty) sequence of cities, $c_1, c_2, \ldots c_m$, such that for each i ($0 \le i < N$) the sequence contains city i exactly $W[i]$ times.

The **delivery time** of a plan $c_1, c_2, \ldots c_m$ is the sum of the distances of consecutive cities in the sequence $0, c_1, c_2, \ldots, c_m, 0$, that is, $d(0, c_1) + d(c_1, c_2) + \cdots + d(c_m, 0)$.

Karcsi has to work for Q days. At the start of each day, the number of required deliveries changes in one of the cities. For some city S and nonnegative integer X, the value of $W[S]$ becomes X. The value of $W[S]$ remains X as long as it is not modified again at the start of a day later on.

Karcsi gets paid by the hour. He wants to choose a delivery plan so that the delivery time is the maximum over all possible plans. Your task is to compute the maximum delivery time for each day when Karcsi has to work.

Implementation Details

Your task is to implement two procedures.

```
void init(int N, int[] U, int[] V, int[] T, int[] W)
```

- N: the number of cities.
- U, V: arrays of length $N-1$ describing road connections.
- T: array of length $N-1$ describing road lengths.
- W: array of length N describing the number of deliveries in each city.
- This procedure is called exactly once for each test case, before any calls to `max_time`.

```
int64 max_time(int S, int X)
```

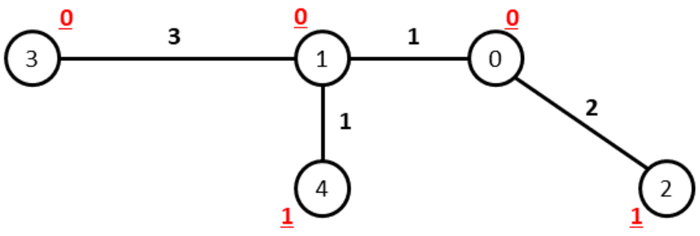
- S, X: integers describing a change in the number of deliveries. The value of $W[S]$ should be set to X.
- This procedure should first perform the specified update, and then return the maximum delivery time over all possible delivery plans.
- This procedure is called exactly Q times.

Example

Consider the following sequence of calls:

```
init(5, [0, 0, 1, 1], [1, 2, 3, 4], [1, 2, 3, 1], [0, 0, 1, 0, 1])
```

The parameters correspond to the road network below. Red numbers represent the initial number of deliveries in each city.



```
max_time(0, 1)
```

After the update, we have $W = [1, 0, 1, 0, 1]$. A possible delivery plan is the sequence $4, 2, 0$. In this case, Karcsi visits cities $0, 4, 2, 0, 0$ in this order, and the delivery time is $d(0,4) + d(4,2) + d(2,0) + d(0,0) = 2 + 4 + 2 + 0 = 8$.

There is no delivery plan with a delivery time greater than 8, thus the procedure should return 8.

Further examples of calls to `max_time` are summarized in the following table.

Call	W after the update	Optimal plan	Maximum delivery time
`max_time(3, 3)`	$[1, 0, 1, 3, 1]$	$3, 0, 3, 2, 3, 4$	$4 + 4 + 4 + 6 + 6 + 4 + 2 = 30$
`max_time(0, 0)`	$[0, 0, 1, 3, 1]$	$3, 2, 3, 4, 3$	$4 + 6 + 6 + 4 + 4 + 4 = 28$
`max_time(4, 0)`	$[0, 0, 1, 3, 0]$	$3, 2, 3, 3$	$4 + 6 + 6 + 0 + 4 = 20$
`max_time(2, 0)`	$[0, 0, 0, 3, 0]$	$3, 3, 3$	$4 + 0 + 0 + 4 = 8$
`max_time(3, 0)`	$[0, 0, 0, 0, 0]$	empty	0

Constraints

- $2 \le N \le 100\,000$
- $0 \le U[j] < V[j] < N$ (for each j such that $0 \le j \le N - 2$)
- $1 \le T[j] \le 100$ (for each j such that $0 \le j \le N - 2$)
- It is possible to travel from any city to any other city by using the roads.
- $0 \le W[i] \le 10^6$ (for each i such that $0 \le i < N$)
- $1 \le Q \le 300\,000$
- $0 \le S < N$
- $0 \le X \le 10^6$

Subtasks

1. (8 points) $N = 2$
2. (21 points) $N, Q \le 1\,000$
3. (14 points) $U[j] = \lfloor \frac{j}{2} \rfloor$ and $V[j] = j + 1$ for each j such that $0 \le j \le N - 2$
4. (21 points) $U[j] = j$ and $V[j] = j + 1$ for each j such that $0 \le j \le N - 2$
5. (36 points) No additional constraints.

Sample grader

The sample grader reads the input in the following format:

- line 1: $N \; Q$
- line 2: $U[0] \; U[1] \; \ldots \; U[N - 2]$
- line 3: $V[0] \; V[1] \; \ldots \; V[N - 2]$
- line 4: $T[0] \; T[1] \; \ldots \; T[N - 2]$
- line 5: $W[0] \; W[1] \; \ldots \; W[N - 1]$
- line $6 + k$ ($0 \le k < Q$): $S \; X$ for update k

The sample grader prints your answers in the following format:

- line $1 + k$ ($0 \leq k < Q$): the return value of `max_time` after update k