Codeforces Round #526 (Div. 2)

A. The Fair Nut and Elevator

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Fair Nut lives in n story house. a_i people live on the i-th floor of the house. Every person uses elevator twice a day: to get from the floor where he/she lives to the ground (first) floor and to get from the first floor to the floor where he/she lives, when he/she comes back home in the evening.

It was decided that elevator, when it is not used, will stay on the x-th floor, but x hasn't been chosen yet. When a person needs to get from floor a to floor b, elevator follows the simple algorithm:

- Moves from the x-th floor (initially it stays on the x-th floor) to the a-th and takes the passenger.
- Moves from the a-th floor to the b-th floor and lets out the passenger (if a equals b, elevator just opens and closes the doors, **but still** comes to the floor from the x-th floor).
- Moves from the b-th floor back to the x-th.

The elevator never transposes more than one person and always goes back to the floor x before transposing a next passenger. The elevator spends one unit of electricity to move between neighboring floors. So moving from the a-th floor to the b-th floor requires $|a - b|$ units of electricity.
Your task is to help Nut to find the minimum number of electricity units, that it would be enough for one day, by choosing an optimal the x-th floor. Don't forget than elevator initially stays on the x-th floor.

Input
The first line contains one integer n ($1 \le n \le 100$) — the number of floors.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 100$) — the number of people on each floor.

Output
In a single line, print the answer to the problem — the minimum number of electricity units.

Examples

input
3 0 2 1
output
16

input
2 1 1
output
4

Note
In the first example, the answer can be achieved by choosing the second floor as the x-th floor. Each person from the second floor (there are two of them) would spend 4 units of electricity per day (2 to get down and 2 to get up), and one person from the third would spend 8 units of electricity per day (4 to get down and 4 to get up). $4 \cdot 2 + 8 \cdot 1 = 16$.

In the second example, the answer can be achieved by choosing the first floor as the x-th floor.

B. Kvass and the Fair Nut

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Fair Nut likes kvass very much. On his birthday parents presented him n kegs of kvass. There are v_i liters of kvass in the i-th keg. Each keg has a lever. You can pour your glass by **exactly** 1 liter pulling this lever. The Fair Nut likes this drink very much, so he wants to pour his glass by s liters of kvass. But he wants to do it, so kvass level in the least keg is as much as possible.

Help him find out how much kvass can be in the least keg or define it's not possible to pour his glass by s liters of kvass.

Input
The first line contains two integers n and s ($1 \le n \le 10^3$, $1 \le s \le 10^{12}$) — the number of kegs and glass volume.

The second line contains n integers $v_1, v_2, \ldots, v_n$ ($1 \le v_i \le 10^9$) — the volume of i-th keg.

Output
If the Fair Nut cannot pour his glass by s liters of kvass, print -1. Otherwise, print a single integer — how much kvass in the least keg can be.

Examples

input
3 3 4 3 5
output
3

input
3 4 5 3 4
output
2

input
3 7 1 2 3
output
-1

Note
In the first example, the answer is 3, the Fair Nut can take 1 liter from the first keg and 2 liters from the third keg. There are 3 liters of kvass in each keg.

In the second example, the answer is 2, the Fair Nut can take 3 liters from the first keg and 1 liter from the second keg.

In the third example, the Fair Nut can't pour his cup by 7 liters, so the answer is -1.

C. The Fair Nut and String

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Fair Nut found a string s. The string consists of lowercase Latin letters. The Nut is a curious guy, so he wants to find the number of strictly increasing sequences $p_1, p_2, \ldots, p_k$, such that:

1. For each i ($1 \le i \le k$), $s_{p_i} =$ 'a'.
2. For each i ($1 \le i < k$), there is such j that $p_i < j < p_{i+1}$ and $s_j =$ 'b'.

The Nut is upset because he doesn't know how to find the number. Help him.

This number should be calculated modulo $10^9 + 7$.

Input
The first line contains the string s ($1 \le |s| \le 10^5$) consisting of lowercase Latin letters.

Output
In a single line print the answer to the problem — the number of such sequences $p_1, p_2, \ldots, p_k$ modulo $10^9 + 7$.

Examples

input
abbaa
output
5

input
baaaa
output
4

input
agaa
output
3

Note
In the first example, there are 5 possible sequences. $[1]$, $[4]$, $[5]$, $[1, 4]$, $[1, 5]$.

In the second example, there are 4 possible sequences. $[2]$, $[3]$, $[4]$, $[5]$.

In the third example, there are 3 possible sequences. $[1]$, $[3]$, $[4]$.

D. The Fair Nut and the Best Path

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Fair Nut is going to travel to the Tree Country, in which there are n cities. Most of the land of this country is covered by forest. Furthermore, the local road system forms a tree (connected graph without cycles). Nut wants to rent a car in the city u and go by a simple path to city v. He hasn't determined the path, so it's time to do it. Note that chosen path can consist of only one vertex.

A filling station is located in every city. Because of strange law, Nut can buy only w_i liters of gasoline in the i-th city. We can assume, that he has **infinite money**. Each road has a length, and as soon as Nut drives through this road, the amount of gasoline decreases by length. Of course, Nut can't choose a path, which consists of roads, where he runs out of gasoline. He can buy gasoline in **every** visited city, even in **the first** and **the last**.

He also wants to find the maximum amount of gasoline that he can have at the end of the path. Help him: count it.

Input
The first line contains a single integer n ($1 \le n \le 3 \cdot 10^5$) — the number of cities.

The second line contains n integers $w_1, w_2, \ldots, w_n$ ($0 \le w_i \le 10^9$) — the maximum amounts of liters of gasoline that Nut can buy in cities.

Each of the next $n - 1$ lines describes road and contains three integers u, v, c ($1 \le u, v \le n$, $1 \le c \le 10^9$, $u \neq v$), where u and v — cities that are connected by this road and c — its length.

It is guaranteed that graph of road connectivity is a tree.

Output
Print one number — the maximum amount of gasoline that he can have at the end of the path.

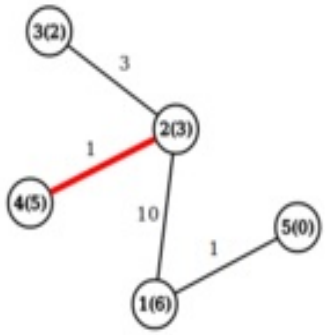
Examples

input
3 1 3 3 1 2 2 1 3 2
output
3

input
5 6 3 2 5 0 1 2 10 2 3 3 2 4 1 1 5 1
output
7

Note
The optimal way in the first example is $2 \to 1 \to 3$.

The optimal way in the second example is $2 \to 4$.



E. The Fair Nut and Strings

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Recently, the Fair Nut has written k strings of length n, consisting of letters "a" and "b". He calculated c — the number of strings that are prefixes of at least one of the written strings. **Every string was counted only one time**.

Then, he lost his sheet with strings. He remembers that all written strings were lexicographically **not smaller** than string s and **not bigger** than string t. He is interested: what is the maximum value of c that he could get.

A string a is lexicographically smaller than a string b if and only if one of the following holds:

- a is a prefix of b, but $a \neq b$;
- in the first position where a and b differ, the string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

Input

The first line contains two integers n and k ($1 \leq n \leq 5 \cdot 10^5$, $1 \leq k \leq 10^9$).

The second line contains a string s ($|s| = n$) — the string consisting of letters "a" and "b.

The third line contains a string t ($|t| = n$) — the string consisting of letters "a" and "b.

It is guaranteed that string s is lexicographically not bigger than t.

Output

Print one number — maximal value of c.

Examples

input
2 4 aa bb
output
6

input
3 3 aba bba
output
8

input
4 5 abbb baaa
output
8

Note

In the first example, Nut could write strings "aa", "ab", "ba", "bb". These 4 strings are prefixes of at least one of the written strings, as well as "a" and "b". Totally, 6 strings.

In the second example, Nut could write strings "aba", "baa", "bba".

In the third example, there are only two different strings that Nut could write. If both of them are written, $c = 8$.

F. Max Mex

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Once Grisha found a tree (connected graph without cycles) with a root in node 1.

But this tree was not just a tree. A permutation p of integers from 0 to $n - 1$ is written in nodes, a number p_i is written in node i.

As Grisha likes to invent some strange and interesting problems for himself, but not always can solve them, you need to help him deal with two types of queries on this tree.

Let's define a function $MEX(S)$, where S is a set of non-negative integers, as a smallest non-negative integer that is not included in this set.

Let l be a simple path in this tree. So let's define indices of nodes which lie on l as u_1, u_2, $\ldots$, u_k.

Define $V(l)$ as a set $\{p_{u_1}, p_{u_2}, \ldots, p_{u_k}\}$.

Then queries are:

1. For two nodes i and j, swap p_i and p_j.
2. Find the maximum value of $MEX(V(l))$ in all possible l.

Input

The first line contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of nodes of a tree.

The second line contains n integers — p_1, p_2, $\ldots$, p_n ($0 \le p_i < n$) — the permutation p, it's guaranteed that all numbers are different .

The third line contains $n - 1$ integers — d_2, d_3, $\ldots$, d_n ($1 \le d_i < i$), where d_i is a direct ancestor of node i in a tree.

The fourth line contains a single integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries.

The following q lines contain the description of queries:

At the beginning of each of next q lines, there is a single integer t (1 or 2) — the type of a query:

1. If $t = 1$, the line also contains two integers i and j ($1 \le i, j \le n$) — the indices of nodes, where values of the permutation should be swapped.
2. If $t = 2$, you need to find the maximum value of $MEX(V(l))$ in all possible l.

Output

For each type 2 query print a single integer — the answer for this query.

Examples

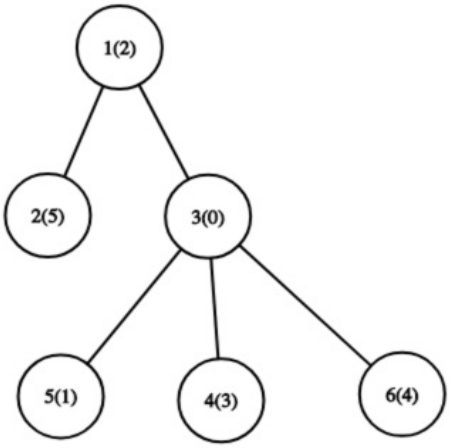
input
6 2 5 0 3 1 4 1 1 3 3 3 3 2 1 6 3 2
output
3 2

input
6 5 2 1 4 3 0 1 1 1 3 3 9 2 1 5 3 2 1 6 1 2 1 4 2 2 1 1 6 2
output
3 2

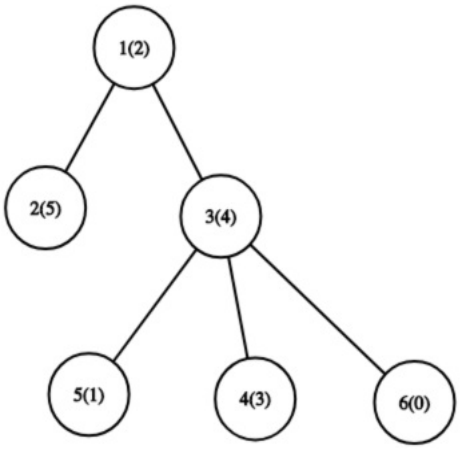
```
4
4
2
```

Note

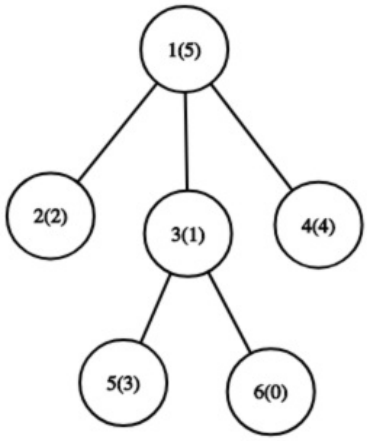
Number written in brackets is a permutation value of a node.



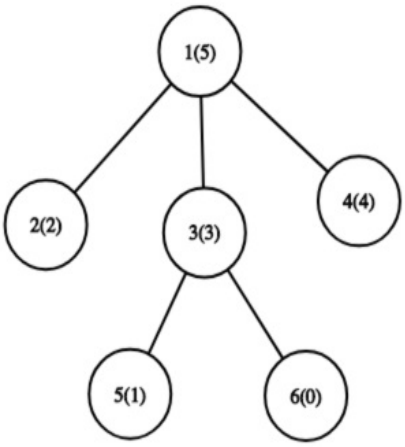
In the first example, for the first query, optimal path is a path from node 1 to node 5. For it, set of values is $\{0, 1, 2\}$ and MEX is 3.



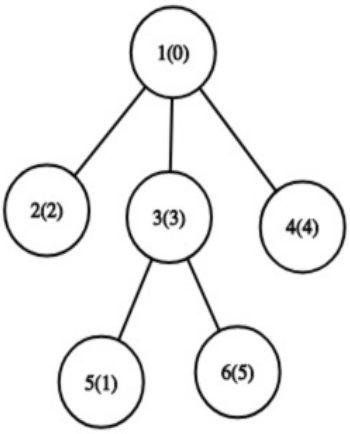
For the third query, optimal path is a path from node 5 to node 6. For it, set of values is $\{0, 1, 4\}$ and MEX is 2.



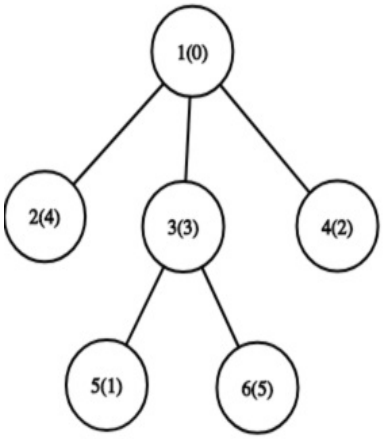
In the second example, for the first query, optimal path is a path from node 2 to node 6. For it, set of values is $\{0, 1, 2, 5\}$ and MEX is 3.



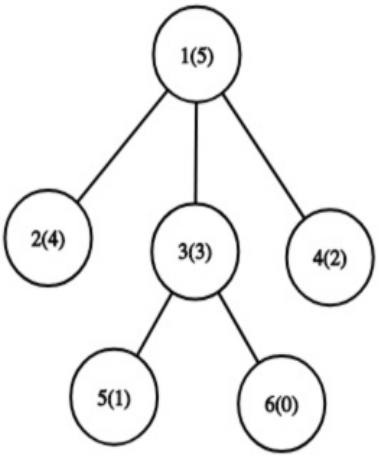
For the third query, optimal path is a path from node 5 to node 6. For it, set of values is $\{0, 1, 3\}$ and MEX is 2.



For the fifth query, optimal path is a path from node 5 to node 2. For it, set of values is $\{0, 1, 2, 3\}$ and MEX is 4.



For the seventh query, optimal path is a path from node 5 to node 4. For it, set of values is $\{0, 1, 2, 3\}$ and MEX is 4.



For the ninth query, optimal path is a path from node 6 to node 5. For it, set of values is $\{0, 1, 3\}$ and MEX is 2.
