## A. Sum in the tree

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Mitya has a rooted tree with $n$ vertices indexed from $1$ to $n$, where the root has index $1$. Each vertex $v$ initially had an integer number $a_v \ge 0$ written on it. For every vertex $v$ Mitya has computed $s_v$: the sum of all values written on the vertices on the path from vertex $v$ to the root, as well as $h_v$ — the depth of vertex $v$, which denotes the number of vertices on the path from vertex $v$ to the root. Clearly, $s_1 = a_1$ and $h_1 = 1$.

Then Mitya erased all numbers $a_v$, and by accident he also erased all values $s_v$ for vertices with even depth (vertices with even $h_v$). Your task is to restore the values $a_v$ for every vertex, or determine that Mitya made a mistake. In case there are multiple ways to restore the values, you're required to find one which minimizes the total sum of values $a_v$ for all vertices in the tree.

### Input

The first line contains one integer $n$ — the number of vertices in the tree ($2 \le n \le 10^5$). The following line contains integers $p_2$, $p_3$, ... $p_n$, where $p_i$ stands for the parent of vertex with index $i$ in the tree ($1 \le p_i < i$). The last line contains integer values $s_1$, $s_2$, ..., $s_n$ ($-1 \le s_v \le 10^9$), where erased values are replaced by $-1$.

### Output

Output one integer — the minimum total sum of all values $a_v$ in the original tree, or $-1$ if such tree does not exist.

### Examples

**input**

```
5
1 1 1 1
1 -1 -1 -1 -1
```

**output**

```
1
```

**input**

```
5
1 2 3 1
1 -1 2 -1 -1
```

**output**

```
2
```

**input**

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

**output**

```
-1
```

## B. Nice table

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

You are given an $n \times m$ table, consisting of characters «A», «G», «C», «T». Let's call a table *nice*, if every $2 \times 2$ square contains all four distinct characters. Your task is to find a nice table (also consisting of «A», «G», «C», «T»), that differs from the given table in the minimum number of characters.

### Input

First line contains two positive integers $n$ and $m$ — number of rows and columns in the table you are given ( $2 \le n, m, n \times m \le 300\,000$). Then, $n$ lines describing the table follow. Each line contains exactly $m$ characters «A», «G», «C», «T».

### Output

Output $n$ lines, $m$ characters each. This table must be nice and differ from the input table in the minimum number of characters.

**Examples**

| input |
|---|
| 2 2<br>AG<br>CT |

| output |
|---|
| AG<br>CT |

| input |
|---|
| 3 5<br>AGCAG<br>AGCAG<br>AGCAG |

| output |
|---|
| TGCAT<br>CATGC<br>TGCAT |

## Note

In the first sample, the table is already nice. In the second sample, you can change 9 elements to make the table nice.

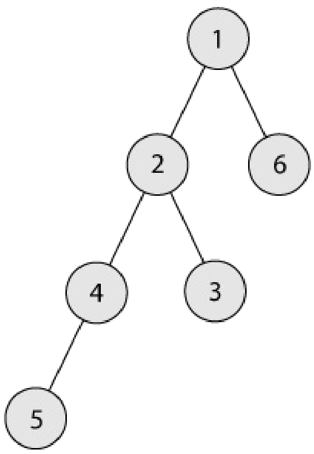
# C. Construct a tree

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

Misha walked through the snowy forest and he was so fascinated by the trees to decide to draw his own tree!

Misha would like to construct a rooted tree with $n$ vertices, indexed from 1 to $n$, where the root has index 1. Every other vertex has a *parent* $p_i$, and $i$ is called a *child* of vertex $p_i$. Vertex $u$ belongs to the *subtree* of vertex $v$ iff $v$ is reachable from $u$ while iterating over the parents ($u$, $p_u$, $p_{p_u}$, ...). Clearly, $v$ belongs to its own subtree, and the number of vertices in the subtree is called the *size* of the subtree. Misha is only interested in trees where every vertex belongs to the subtree of vertex $1$.

Below there is a tree with $6$ vertices. The subtree of vertex $2$ contains vertices $2$, $3$, $4$, $5$. Hence the size of its subtree is $4$.




The *branching coefficient* of the tree is defined as the maximum number of children in any vertex. For example, for the tree above the branching coefficient equals $2$. Your task is to construct a tree with $n$ vertices such that the sum of the subtree sizes for all vertices equals $s$, and the branching coefficient is minimum possible.

## Input

The only input line contains two integers $n$ and $s$ — the number of vertices in the tree and the desired sum of the subtree sizes ( $2 \le n \le 10^5$; $1 \le s \le 10^{10}$).

## Output

If the required tree does not exist, output «No». Otherwise output «Yes» on the first line, and in the next one output integers $p_2$, $p_3$, ..., $p_n$, where $p_i$ denotes the parent of vertex $i$.

**Examples**

| input |
|---|
| 3 5 |

**output**

```
Yes
1 1
```

**input**

```
4 42
```

**output**

```
No
```

**input**

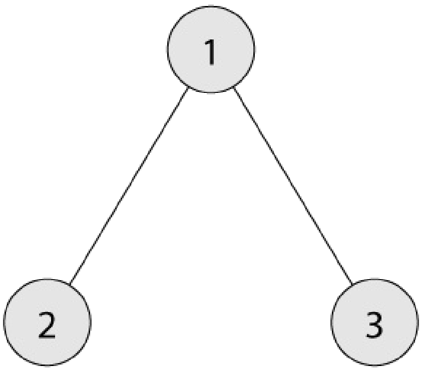
```
6 15
```

**output**

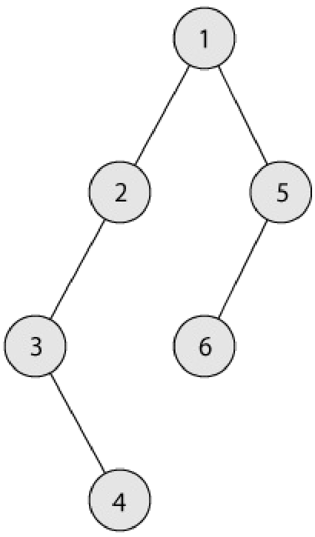
```
Yes
1 2 3 1 5
```

### Note

Below one can find one of the possible solutions for the first sample case. The sum of subtree sizes equals $3+1+1=5$, and the branching coefficient equals $2$.




Below one can find one of the possible solutions for the third sample case. The sum of subtree sizes equals $6+3+2+1+2+1=15$, and the branching coefficient equals $2$.




## D. Eels

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

Vasya is a big fish lover, and his parents gave him an aquarium for the New Year. Vasya does not have a degree in ichthyology, so he thinks that filling a new aquarium with eels is a good idea. Unfortunately, eels are predators, so Vasya decided to find out how dangerous this idea was.

Getting into one aquarium, eels fight each other until exactly one fish remains. When two eels fight, the big one eats the smaller one (if their weights are equal, then one of them will still eat the other). Namely, let $n$ eels be initially in an aquarium, and the $i$-th of them have a weight of $x_i$. Then $n-1$ battles will occur between them, as a result of which, only one eel will survive. In a battle of two eels with weights $a$ and $b$, where $a \le b$, eel of weight $a$ will be eaten and disappear from the aquarium, and eel of weight $b$ will increase its weight to $a+b$.

A battle between two eels with weights $a$ and $b$, where $a \le b$, is considered *dangerous* if $b \le 2a$. For a given set of eels, *danger* is defined as the maximum number of dangerous battles that can occur among these eels if they are placed in one aquarium.

Now Vasya is planning, which eels he wants to put into an aquarium. He has some set of eels (initially empty). He makes a series of operations with this set. With each operation, he either adds one eel in the set, or removes one eel from the set. Vasya asks you to calculate the danger of the current set of eels after each operation.

### Input

The first line of input contains a single integer $q$ ($1 \le q \le 500\,000$), the number of operations that Vasya makes. The next $q$ lines describe operations. Each operation has one of two types :

- `+ x` describes the addition of one eel of weight $x$ to the set ($1 \le x \le 10^9$). Note that in the set there can be several eels of the same weight.
- `- x` describes the removal of one eel of weight $x$ from a set, and it is guaranteed that there is a eel of such weight in the set.

### Output

For each operation, output single integer, the danger of the set of eels after this operation.

#### Examples

**input**

```
2
+ 1
- 1
```

**output**

```
0
0
```

**input**

```
4
+ 1
+ 3
+ 7
- 3
```

**output**

```
0
0
1
0
```

**input**

```
9
+ 2
+ 2
+ 12
- 2
- 2
+ 4
+ 1
+ 1
- 12
```

**output**

```
0
1
1
0
0
0
0
3
2
```

### Note

In the third example, after performing all the operations, the set of eels looks like $\{1, 1, 4\}$. For this set of eels, there are several possible scenarios, if all of them are placed in one aquarium:

- The eel of weight 4 eats the eel of weight 1, and then the second eel of weight 1. In this case, none of the battles are dangerous.
- The eel of weight 1 eats the eel of weight 1, and this battle is dangerous. Now there are two eels in the aquarium, their weights are 4 and 2. The big one eats the small one, and this battle is also dangerous. In this case, the total number of dangerous battles

will be 2.

Thus, the danger of this set of eels is 2.

# E. Fedya the Potter

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Fedya loves problems involving data structures. Especially ones about different queries on subsegments. Fedya had a nice array $a_1, a_2, \ldots a_n$ and a beautiful data structure. This data structure, given $l$ and $r$, $1 \le l \le r \le n$, could find the greatest integer $d$, such that $d$ divides each of $a_l$, $a_{l+1}$, ..., $a_r$.

Fedya really likes this data structure, so he applied it to every non-empty contiguous subarray of array $a$, put all answers into the array and sorted it. He called this array $b$. It's easy to see that array $b$ contains $n(n+1)/2$ elements.

After that, Fedya implemented another cool data structure, that allowed him to find sum $b_l + b_{l+1} + \ldots + b_r$ for given $l$ and $r$, $1 \le l \le r \le n(n+1)/2$. Surely, Fedya applied this data structure to every contiguous subarray of array $b$, called the result $c$ and sorted it. Help Fedya find the lower median of array $c$.

Recall that for a sorted array of length $k$ the *lower median* is an element at position $\lfloor \frac{k+1}{2} \rfloor$, if elements of the array are enumerated starting from $1$. For example, the lower median of array $(1, 1, 2, 3, 6)$ is $2$, and the lower median of $(0, 17, 23, 96)$ is $17$.

### Input

First line contains a single integer $n$ — number of elements in array $a$ ($1 \le n \le 50\,000$).

Second line contains $n$ integers $a_1, a_2, \ldots, a_n$ — elements of the array ($1 \le a_i \le 100\,000$).

### Output

Print a single integer — the lower median of array $c$.

#### Examples

**input**

```
2
6 3
```

**output**

```
6
```

**input**

```
2
8 8
```

**output**

```
8
```

**input**

```
5
19 16 2 12 15
```

**output**

```
12
```

### Note

In the first sample array $b$ is equal to $3, 3, 6$, then array $c$ is equal to $3, 3, 6, 6, 9, 12$, so the lower median is $6$.

In the second sample $b$ is $8, 8, 8$, $c$ is $8, 8, 8, 16, 16, 24$, so the lower median is $8$.

# F. Ж-function

time limit per test: 6 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

The length of the longest common prefix of two strings $s = s_1 s_2 \ldots s_n$ and $t = t_1 t_2 \ldots t_m$ is defined as the maximum $k \le \min(n, m)$ such that $s_1 s_2 \ldots s_k$ equals $t_1 t_2 \ldots t_k$. Let's denote the longest common prefix of two strings $s$ and $t$ as $lcp(s, t)$.

Z-function of a string $s_1 s_2 \ldots s_n$ is a sequence of integers $z_1, z_2, \ldots, z_n$, where $z_i = lcp(s_1 s_2 \ldots s_n,\ \ s_i s_{i+1} \ldots s_n)$. *Ж-function* of a string $s$ is defined as $z_1 + z_2 + \ldots + z_n$.

You're given a string $s = s_1 s_2 \ldots s_n$ and $q$ queries. Each query is described by two integers $l_i$ and $r_i$, where $1 \le l_i \le r_i \le n$. The answer for the query is defined as Ж-function of the string $s_{l_i} s_{l_i+1} \ldots s_{r_i}$.

## Input

The first line contains the string $s$, consisting of lowercase English letters ($1 \le |s| \le 200\,000$). The second line contains one integer $q$ — the number of queries ($1 \le q \le 200\,000$). Each of the following $q$ lines contains two integers $l_i$ and $r_i$, describing the query ( $1 \le l_i \le r_i \le |s|$).

## Output

For every query output one integer: the value of Ж-function of the corresponding substring.

### Examples

**input**

```
abbd
4
2 3
1 3
3 3
1 4
```

**output**

```
3
3
1
4
```

**input**

```
bbaaa
5
2 4
1 5
1 5
3 3
1 2
```

**output**

```
3
6
6
1
3
```

## Note

In the first sample case there are four queries:

- the first query corresponds to the substring bb, and its Ж-function equals $2 + 1 = 3$;
- the second query corresponds to the substring abb, and its Ж-function equals $3 + 0 + 0 = 3$;
- the third query corresponds to the substring b, and its Ж-function equals $1$.
- the fourth query corresponds to the substring abdd, and its Ж-function equals $4 + 0 + 0 + 0 = 4$.