Day 1. SQRT decomposition

A. Range Sum Query Strikes Back

2 seconds, 256 megabytes

You are given an array a consisting of n integers and q queries of two types:

1. set the value of a_p to the value x ($a_p := x$);
2. find the sum inside the segment $[l; r]$ of the array a.

Your task is to answer all queries of the second type.

Solve this problem using sqrt-decomposition.

Input

The first line of the input contains two integers n and q ($1 \le n \le 2 \cdot 10^5; 1 \le q \le 10^5$) — the number of elements in a and the number of queries, respectively.

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the i-th element of the array a.

The next q lines describe queries. The i-th query starts with an integer t_i ($1 \le t_i \le 2$) — the type of the query. Then, if $t_i = 1$, two integers p_i and x_i follow ($1 \le p_i \le n; 1 \le x_i \le 10^9$), where p_i is the position of the element and x_i is the new value of the element at the position p_i. And if $t_i = 2$, then two integers l_i and r_i follow ($1 \le l_i \le r_i \le n$) — borders of the segment you have to find the sum at.

It is guaranteed that there is at least one query of the second type.

Output

On each query of the second type, print one integer — the sum inside the segment $[l; r]$ of the array a (where l and r are integers from the query).

input
5 7 2 5 6 4 1 2 1 3 1 3 1 2 3 5 2 1 5 1 4 100 2 1 4 2 5 5

output
13 6 13 108 1

B. Lesser Elements with Insert

2 seconds, 256 megabytes

You are given q queries of two types:

1. Add one occurrence of x to the array;
2. find the number of elements less than x in the array.

Your task is to answer all queries of the second type.

Solve this problem using sqrt-decomposition.

Input

The first line of the input contains one integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries.

The next q lines describe queries. The i-th line consists of two integers t_i ($1 \le t_i \le 2$) and x_i ($1 \le x_i \le 10^9$) — the type of the query and the element you have to perform the query with.

It is guaranteed that there is at least one query of the second type.

Output

On each query of the second type, print one integer — the number of elements less than x (where x is the integer from the query) in the array.

input
13 2 12 1 4 1 6 2 4 1 5 1 5 2 6 1 8 1 6 1 2 2 7 1 6 2 11

output
0 0 3 6 8

C. Sum Problem

5 seconds, 256 megabytes

Implement a data structure that supports the set of S integers with which the following operations are allowed:

- $\mathrm{add}(i)$ — add the number i to the set S (if it is already there, the set does not change);
- $\mathrm{sum}(l, r)$ — output the sum of all elements x from S that satisfy the inequality $l \le x \le r$.

Input

Initially, the set S is empty. The first line of the input file contains n — the number of operations ($1 \le n \le 300\,000$). The next n lines contain operations. Each operation has the form either "+ i" or "? l r". Operation "? l r" sets the query to $\mathrm{sum}(l, r)$.

If the operation "+ i" is in the input file at the beginning or after another operation "+", then it defines the operation $\mathrm{add}(i)$. If it goes after the query "?", and the result of this query was y, then the operation $\mathrm{add}((i + y) \bmod 10^9)$ is performed.

In all queries and adding operations, the parameters are in the range from 0 to 10^9.

Output

For each request print one number — the response to the request.

input
6 + 1 + 3 + 3 ? 2 4 + 1 ? 2 4

output
3 7

D. Fabrosaurs

4.0 s, 256 megabytes

Fabrosaurs are known for their delicate artistic taste and passion for landscape design. They live near a very picturesque river and now and then rebuild the path that runs along the river: either pour additional land, or tear down what is. In order to simplify this work, they divided the entire path into horizontal sections numbered from 1 to N, and their alterations are always the same: they choose a part of the road from the L th to the R th section (inclusive) and change (increase or decrease) the height in all these areas by the same amount (if before the start of the alteration the heights were different, then after the alteration they will remain different).

Since, as already mentioned, fabrosaurs have a fine artistic taste, each of them believes that their river looks best from a certain height. Therefore, they want to know if there is a place on the path near their home where the height in their opinion is optimal. Help them figure it out.

Input

The first line of the input file contains two numbers N and M — the length of the road and the number of requests, respectively ($1 \le N, M \le 10^5$). The second line contains N numbers, separated by spaces — the initial heights of the corresponding parts of the road; heights do not exceed 10^4 in absolute value. The next M lines contain requests one per line.

The query $+ \ L \ R \ X$ means that the height of the parts of the road from L th to R th (inclusive) needs to be changed to X. Moreover, $1 \le L \le R \le N$, and $|X| \le 10^4$.

The query $? \ L \ R \ X$ means that you need to check if there is a section between the L and R sections (including these sections) where the road passes exactly at a height of X. It is guaranteed that $1 \le L \le R \le N$, and $|X| \le 10^9$.

Output

For each request of the second type, you need to output one word « texttt {YES} » (without quotes) to the output file on a separate line if the necessary section exists, and « texttt {NO} » otherwise.

input
10 5 0 1 1 3 3 3 2 0 0 1 ? 3 5 2 + 1 4 1 ? 3 5 2 + 7 10 2 ? 9 10 3

output
NO YES YES

E. Madness and Courage

10 seconds, 512 megabytes

Many of us have been dreaming of becoming a game developer. For some of us this even has become the reason to start studying computer science. And one day the dream came true for lucky Michael — he works as a software engineer in a top company «Snowstorm», which is famous for it's well-known masterpieces, such as «Military Art» and «Stellar Job».

Recently Michael has joined a group that works on a new astonishing role playing game «Madness and Courage». Its main feature is the possibility for the player to choose a new character at the beginning of every level.

Before a new level starts, there are N heroes available to choose from. Each hero is characterized by the amount of strength of his attack points a_i and the amount of initial health points b_i. The level is a long cave with M monsters waiting inside. Each monster also has strength of attack c_i and initial healh points d_i. The chosen heroe enters the cave and starts to fight with the first monster. If he survives in the battle, he fights against the second one, then the third and so on, until he dies or the level ends. Health points are not restored between the fights, that means the hero always starts a new fight with the smaller amount of health points, than in the previous one.

The hero versus monster fight constists of simultaneous strikes they give to each other. Each strike decreases the amount of the enemy's health points by the value of the striker's strength of attack. As soon as anyone's amount of health points turns non-positive, he dies and the fight ends. Please note, that according to these fight rules, it can happen that both due hero and monster will die simultaneously.

The company plans to make the game free and obtain money from selling different hints and additional content. Michael is going to create the hint, that tells every hero, how many monsters this hero will kill, if he is chosen by the player to fight at this level. As there may be an enormous number of heroes and monsters, Michael needs your help to compute the hint values.

Input

First line of the input contains two integers N and M — the number of heroes player can choose from and the number of monsters waiting in the cave respectively ($1 \le N, M \le 200\,000$).

Each of next N lines contain hero description. For every hero two integer numbers a_i and b_i are given, they correspond to the hero's strength of attack and initial amount of health points ($1 \le a_i, b_i \le 10^9$).

Then follow M lines, each of them describing a single monster in the cave. For every monster values c_i and d_i are given, corresponding to his strength of attack and amount of health points ($1 \le c_i, d_i \le 200\,000$). Monsters in the cave are situated in the same order as they appear in the input. That means, right after the hero enters the cave, he needs to kill the monster described in $N+2$ line of the input. Right before the end of the level due hero will fight against the monster described in the line $N+M+1$ of the input file.

Output

Print N lines containing one integer number each. On the i-th line print the answer for the i-th hero.

input

```
5 3
1 2
2 2
10 10
100 10
1 100
2 2
7 2
3 20
```

output

```
0
1
2
3
3
```

First hero versus first monster fight will last for only one round, as the hero will be dead after it and the monster will be still alive with one health point left.

Second hero has absolutely the same parameters as the first monster. This means they will kill each other, so the answer for this hero is one.

If the player chooses the third hero to go through the cave, eight health points will be left after fighting due first monster, and only one health point after killing the second one. This hero need to give two strikes to kill third monster, but he will die after monster's first strike.

Fourth hero has the same amount of health points as the third, but has much greater strength of attack, so he will kill all the monsters, but will also die at the end of the fight with the last monster.

Fifth hero has minimal possible strength of attack, but is durable because of the high number of initial health points. He is the only hero to kill all the monsters and stay alive. After fighting first monster he will have 96 health points left, after fighting with due second — 82, and only 22 after killing the last monster.

F. Candies Eating

2 seconds, 256 megabytes

There are n piles of candies in a row. The i-th pile contains a_i candies.

You are given q queries. The i-th query consists of three integers l_i, r_i and x_i — eat x_i candies from all piles with indices between l_i and r_i. If some pile contains less than x_i candies, you just eat all the remaining candies from this pile.

Your task is to say for each pile **the index of query** after which this pile becomes empty (or -1 if the pile will not be empty after all queries).

Input

The first line of the input contains two integers n and q ($1 \le n, q \le 10^5$) — the number of piles of candies and the number of queries, respectively.

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the number of candies in the i-th pile.

The next q lines describe queries. The i-th query consists of three integers l_i, r_i and x_i ($1 \le l_i \le r_i \le n; 1 \le x_i \le 10^9$), where l_i and r_i are borders of the segment and x_i is the number of candies you have to eat.

Output

Print n integers — for each pile in order of the input print **the index of query** after which this pile becomes empty (or -1 if the pile will not be empty after all queries).

input

```
2 2
1 1
1 1 1
1 2 1
```

output

```
1 2
```

input

```
5 7
10 3 3 7 1
1 2 1
1 5 1
1 4 4
1 5 2
1 5 4
1 2 4
3 3 3
```

output

```
5 3 3 4 2
```

input

```
7 6
2 4 4 3 7 10 5
1 2 1
3 6 4
2 4 2
4 4 5
1 4 1
1 7 5
```

output

```
5 5 2 2 6 -1 6
```

G. Little Elephant and Array

4 seconds, 256 megabytes

The Little Elephant loves playing with arrays. He has array a, consisting of n positive integers, indexed from 1 to n. Let's denote the number with index i as a_i.

Additionally the Little Elephant has m queries to the array, each query is characterised by a pair of integers l_j and r_j ($1 \le l_j \le r_j \le n$). For each query l_j, r_j the Little Elephant has to count, how many numbers x exist, such that number x occurs exactly x times among numbers $a_{l_j}, a_{l_j+1}, \ldots, a_{r_j}$.

Help the Little Elephant to count the answers to all queries.

Input

The first line contains two space-separated integers n and m ($1 \le n, m \le 10^5$) — the size of array a and the number of queries to it. The next line contains n space-separated positive integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$). Next m lines contain descriptions of queries, one per line. The j-th of these lines contains the description of the j-th query as two space-separated integers l_j and r_j ($1 \le l_j \le r_j \le n$).

Output

In m lines print m integers — the answers to the queries. The j-th line should contain the answer to the j-th query.

input

```
7 2
3 1 2 2 3 3 7
1 7
3 4
```

output

```
3
1
```

H. Powerful array

4 seconds, 256 megabytes

An array of positive integers $a_1, a_2, ..., a_n$ is given. Let us consider its arbitrary subarray $a_l, a_{l+1}..., a_r$, where $1 \le l \le r \le n$. For every positive integer s denote by K_s the number of occurrences of s into the subarray. We call the *power* of the subarray the sum of products $K_s \cdot K_s \cdot s$ for every positive integer s. The sum contains only finite number of nonzero summands as the number of different values in the array is indeed finite.

You should calculate the power of t given subarrays.

Input

First line contains two integers n and t ($1 \le n, t \le 200000$) — the array length and the number of queries correspondingly.

Second line contains n positive integers a_i ($1 \le a_i \le 10^6$) — the elements of the array.

Next t lines contain two positive integers l, r ($1 \le l \le r \le n$) each — the indices of the left and the right ends of the corresponding subarray.

Output

Output t lines, the i-th line of the output should contain single positive integer — the power of the i-th query subarray.

Please, do not use `%lld` specificator to read or write 64-bit integers in C++. It is preferred to use `cout` stream (also you may use `%I64d`).

input

```
3 2
4 7 4
2 3
1 3
```

output

```
11
23
```

input

```
8 4
3 3 6 6 3 7 3 3
2 3
1 4
2 2
1 8
```

output

```
9
36
3
106
```

Consider the following array (see the second sample) and its [2, 7] subarray (elements of the subarray are colored):

Then $K_1 = 3, K_2 = 2, K_3 = 1$, so the power is equal to $3^2 \cdot 1 + 2^2 \cdot 2 + 1^2 \cdot 3 = 20$.