Contents

Warm-up

Problem A. The Sum [0.5 sec]

You are given an array of N elements.
You have to calculate sum of elements on the segment several times.

Input

The first line contains two integers N and K — length of the array and number of queries. $(1 \leqslant N \leqslant 100\,000)$, $(0 \leqslant K \leqslant 100\,000)$. The next K lines contain queries

1. `A i x` — set i-th element of the array equal to x $(1 \leqslant i \leqslant n,\ 0 \leqslant x \leqslant 10^9)$.
2. `Q l r` — calculate sum of all numbers on positions from l to r $(1 \leqslant l \leqslant r \leqslant n)$.

Initially array contains N zeroes.

Output

For each query of type "`Q l r`" output unique integer number — the sum.

Examples

sum.in	sum.out
5 9 A 2 2 A 3 1 A 4 2 Q 1 1 Q 2 2 Q 3 3 Q 4 4 Q 5 5 Q 1 5	0 2 1 2 0 5

Tutorial

Solved in the lection.

Easy Problems

Problem B. Fabrozavrs-designers [1.1 sec]

You are given an array of N elements. You have to perform queries
`"+= [L,R]"`, `"bool containsIn(x, [L..R])"`.

Input

The first line contains two integers N and M — length of the array and number of queries ($1 \leqslant N, M \leqslant 10^5$). The second line contains N integers, divided by spaces — initial numbers in the array. All numbers do not exceed 10^4 by absolute value. Next M lines contain queries. One per line.

Query `"+ L R X"` means, numbers from L to R should be increased by X. $1 \leqslant L \leqslant R \leqslant N$, and $|X| \leqslant 10^4$.

Query `"? L R X"` means, you should check is there any number eqaul to X on the segment $[L, R]$. It's garantued $1 \leqslant L \leqslant R \leqslant N$, and $|X| \leqslant 10^9$.

Output

For each query of the second type output «`YES`» (without quotes), the number X meets on the segment $[L, R]$, and «`NO`» in the other case.

Examples

fabro.in	fabro.out
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	NO YES YES

Tutorial

SQRT decomposition on array.

$[L, R]$ = head + body + tail. Process head and tail by hands in time $\mathcal{O}(\sqrt{n})$. For every block you may store sorted array or hash table. Also you should store `add[block]` – value to add to all numbers in the block.

Problem C. Count Online [4.8 sec]

You are given multiset of points on the plane.

You have to perform queries of two types:

∘ "? x_1 y_1 x_2 y_2" — how many points are in $[x_1..x_2] \times [y_1..y_2]$?
Notice, points on the border and in the corner are also in. $x_1 \leqslant x_2$, $y_1 \leqslant y_2$.

∘ "+ x y" — add to the multiset the point `(x + res % 100, y + res % 101)`. Here `res` is the answer to the last query of type `?`, and `%` — residue modulo.

Input

Number of points N ($1 \leqslant N \leqslant 50\,000$). Then N points. Then number of queries Q ($1 \leqslant Q \leqslant 10^5$). Then Q queries. All coordinares are from 0 to 10^9.

Output

For each query of type "?" output one integer — number of points in the rectangle.

Example

countonline.in	countonline.out
5 0 0 1 0 0 1 1 1 1 1 9 ? 0 1 1 2 + 1 2 + 2 2 ? 1 0 2 2 ? 0 0 0 0 + 3 3 ? 3 3 3 3 ? 4 3 4 3 ? 4 4 5 5	3 3 1 0 0 3

Note

Tutorial

SQRT decomposition on queries.

To solve the problem without new points, lets use "*range tree of sorted arrays*".

Lets rebuild our structure each $\sqrt{n \log n}$ time.

`time(add)` $= \sqrt{n \log n}$

`time(get)` $= \sqrt{n \log n} + \log^2 n$

Problem D. Substring Query [1.4 sec]

Bobo has n strings $S_1, S_2, \ldots, S_n$. One day, his friend yiyi comes and asks him q questions: how many strings in $S_{l_i}, S_{l_i+1}, \ldots, S_{r_i}$ containing P_i as a substring?

Help Bobo find out the answer.

Input

The first line contains two integers n, q ($1 \le n, q \le 200\,000$).

Each of the following n lines contains 1 string S_i ($|S_1| + |S_2| + \cdots + |S_n| \le 200\,000$).

Each of the last q lines contains 2 integers l_i, r_i and string P_i.

$(1 \le l_i \le r_i \le n, |P_1| + |P_2| + \cdots + |P_n| \le 200\,000)$

All strings consist of "a" and "b".

Output

For each question output single integer, which denotes the answer.

Examples

str-qry.in	str-qry.out
4 2 a b ab bab 1 3 a 1 4 ab	2 2

Tutorial

SQRT decomposition on strings.

Lets iterate length of P_i. For each length the solution is:

1. Precalculate in $\mathcal{O}(|S_1| + \cdots + |S_n|)$ hash table `v[hash]`.
2. `v[h]` – sorted vector of indeces i of S_i that has substring with hash equal to `h`.
3. Query: two binary searches in `v[getHash(P`$_j$`)]`.

Not so easy

Problem E. Yet another k-th statistic [5 sec]

Initially you have an array of integer numbers.

You have to perform three types of queries:

- `+ i x` — Insert the number x to the i-th position (size of the array increases by one)
- `- i` — Erase the number on i-th position of array (size of the array decreases by one)
- `? L R x` — Say, how many numbers y on positions $L \leqslant i \leqslant R$ such, that $y \leqslant x$ ($|x| \leqslant 10^9$)

All indeces i, L, R are numbered from zero. All numbers in queries are integer. All queries are correct. Example of the query: `"+ 0 x"` means "insert x to the beginning of the array". Initially amount of elements in array is N ($0 \leqslant N \leqslant 10^5$). Numbers in array do not exceed 10^9 by absolute value. Amount of queries is K ($1 \leqslant K \leqslant 10^5$).

Example

kthstat.in	kthstat.out
10 455184306 359222813 948543704 914773487 861885581 253523 770029097 193773919 581789266 457415808 - 1 ? 2 5 527021001 ? 0 5 490779085 ? 0 5 722862778 + 9 448694272 - 5 ? 1 2 285404014 - 4 ? 3 4 993634734 + 0 414639071	1 2 2 0 2

Tutorial

Solved in the lection.

Short description:
SQRT decomposition with split & rebuild; for each block lets store "*sorted array*".

Problem F. Tin-plate [4.3 sec]

You are given an array of N integers. You have to perform queries of three types:

- `get(L, R, x)` — calculate amount of elements in $[L..R]$ that are not less than x.
- `set(L, R, x)` — set to all elements in $[L..R]$ value x.
- `reverse(L, R)` — reverse segment $[L..R]$.

Input

The first line contains number N ($1 \leqslant N \leqslant 10^5$). The second line contains array of N integers. Then number of queries M ($1 \leqslant M \leqslant 10^5$) follows. Accurate format of queries you may gather from the sample. All segments in queries satisfy $1 \leqslant L \leqslant R \leqslant N$. All numbers in initial array and all new values are integers from 0 to 10^9.

Output

For each query of type "`get`" output the answer.

Example

sqrtrev.in	sqrtrev.out
5 1 2 3 4 5 6 get 1 5 3 set 2 4 2 get 1 5 3 reverse 1 2 get 2 5 2 get 1 1 2	3 1 3 1

Tutorial

Solved in the lection.

Short description:
SQRT decomposition with split & rebuild; for each block lets store "*sorted array*".

Problem G. Points in halfplane [0.9 sec]

You are given N points on the plane. Points have integer coordinates and are uniformly distributed inside square $[0..C] \times [0..C]$. You have to perform queries kind of "*how many points are in the halfplane?*".

Input

Number of points N ($1 \leqslant N \leqslant 50\,000$), number of queries M ($1 \leqslant M \leqslant 50\,000$), integer constant C ($1 \leqslant C \leqslant 10^4$). Then N points (X_i, Y_i) with integer coordinates from 0 to C. Then M halfplanes (a, b, c). Numbers a, b, c are integers from -10^4 to 10^4. $a^2 + b^2 \neq 0$. We say, point is inside halfplane iff $ax + by + c \geqslant 0$.

Output

For each of M queries output the only integer — number of points inside the halfplane.

Example

semiplane.in	semiplane.out
3 4 10 5 5 1 7 7 4 1 1 -9 1 1 -10 1 1 -11 1 1 -12	2 2 1 0

Tutorial

SQRT decomposition on plane. Lets split our grid $[0..C] \times [0..C]$ into cells of size $k \times k$, here $k = \lceil \frac{C}{\sqrt{n}} \rceil$. We have 2D array A of these cells. In each cell there are $\mathcal{O}(1)$ **different** points. For each cell we know amount of points inside. The solution:

1. Precalculate partial 2D sums for array A.
2. Query. Let $|a| \leqslant |b|$, so the line is more vertical than horizontal.
3. Query. For each row of A there are at most two cells, which are intersected by the line.
4. Query. In $\mathcal{O}(1)$ get sum of all cells except these two; iterate all points in these two cells in $\mathcal{O}$(`numberOfPoints`) time. Do not forget, `numberOfPoints` $= \mathcal{O}(1)$.

Be careful with $C = \mathcal{O}(1)$.
In this case you should compress pack of k equal points into triple $\langle x, y, k \rangle$.