Topic 3: Range Query: Review, Segment Tree Hard

Handout

CS 41100 - CP3 Competitive Programming III (Spring 2024)

Purdue University

Instructor: Zhongtang Luo

Date: February 3, 2025

Sample Problem: Movie Collection
Link: `https://vjudge.net/problem/Gym-100729C`

Sample Problem: Wine Factory (Hard Version)
Link: `https://vjudge.net/problem/CodeForces-1919F2`

Sample Problem: Editor
Link: `https://vjudge.net/problem/CodeForces-1263E`

Sample Problem: Paimon Segment Tree
Link: `https://vjudge.net/problem/Gym-103470E`

Movie Collection

Mr. K. I. has a very large movie collection, organized in a big stack. He tracks the position of each movie by the number of movies above it in the stack. Each movie is identified by a number on its box. When he watches a movie, he removes it from the stack (keeping track of how many movies were above it) and then places it back on top after watching.

Since the stack of movies is so big, he needs to keep track of the position of each movie. It is sufficient to know for each movie how many movies are placed above it, since, with this information, its position in the stack can be calculated. Each movie is identified by a number printed on the movie box.

Your task is to implement a program which will keep track of the position of each movie. In particular, each time Mr. K. I. removes a movie box from the stack, your program should print the number of movies that were placed above it before it was removed.

Input

The first line contains an integer: the number of test cases, at most 100. For each test case:

- One line with two integers m and r ($1 \le m, r \le 100,000$): the number of movies in the stack and the number of locate requests.
- One line with r integers $a_1, \dots, a_r$ ($1 \le a_i \le m$), representing the IDs of movies Mr. K. I. wants to watch.

Initially, the stack contains movies labeled $1, 2, \dots, m$ in increasing order, with movie 1 at the top.

Output

For each test case, output one line with r integers. The i-th integer should give the number of movie boxes above the box labeled a_i, immediately before it is removed from the stack.

Examples

Input

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

Output

```
2 1 0
3 0 4
```

Note

After each locate request a_i, the movie with label a_i is placed on top of the stack.

Source

Northwestern Europe Regional Contest (NWERC) 2011

Wine Factory (Hard Version)

There are three arrays a, b and c. a and b have length n and c has length $n-1$. Let $W(a, b, c)$ denote the liters of wine created from the following process.

Create n water towers. The i-th water tower initially has a_i liters of water and has a wizard with power b_i in front of it. Furthermore, for each $1 \le i \le n-1$, there is a valve connecting water tower i to $i+1$ with capacity c_i.

For each i from 1 to n in this order, the following happens:

1. The wizard in front of water tower i removes at most b_i liters of water from the tower and turns the removed water into wine.
2. If $i \ne n$, at most c_i liters of the remaining water left in water tower i flows through the valve into water tower $i+1$.

There are q updates. In each update, you will be given integers p, x, y and z and you will update $a_p := x$, $b_p := y$ and $c_p := z$. After each update, find the value of $W(a, b, c)$. Note that previous updates to arrays a, b and c persist throughout future updates.

Input

The first line contains two integers n and q ($2 \le n \le 5 \cdot 10^5$, $1 \le q \le 5 \cdot 10^5$) — the number of water towers and the number of updates.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — the number of liters of water in water tower i.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 10^9$) — the power of the wizard in front of water tower i.

The fourth line contains $n-1$ integers $c_1, c_2, \ldots, c_{n-1}$ ($0 \le c_i \le 10^{18}$) — the capacity of the pipe connecting water tower i to $i+1$.

Each of the next q lines contains four integers p, x, y and z ($1 \le p \le n$, $0 \le x, y \le 10^9$, $0 \le z \le 10^{18}$) — the updates done to arrays a, b and c.

Note that c_n does not exist, so the value of z does not matter when $p = n$.

Output

Print q lines, each line containing a single integer representing $W(a, b, c)$ after each update.

Examples

Input

```
4 3
3 3 3 3
1 4 2 8
5 2 1
4 3 8 1000000000
2 5 1 1
3 0 0 0
```

Output

```
11
8
5
```

Input

```
5 5
10 3 8 9 2
3 4 10 8 1
6 5 9 2
5 4 9 1
1 1 1 1
2 7 4 8
4 1 1 1
1 8 3 3
```

Output

```
31
25
29
21
23
```

Note

The first update does not make any modifications to the arrays.

- When $i = 1$, there are 3 liters of water in tower 1 and 1 liter of water is turned into wine. The remaining 2 liters of water flow into tower 2.
- When $i = 2$, there are 5 liters of water in tower 2 and 4 liters of water is turned into wine. The remaining 1 liter of water flows into tower 3.
- When $i = 3$, there are 4 liters of water in tower 3 and 2 liters of water is turned into wine. Even though there are 2 liters of water remaining, only 1 liter of water can flow into tower 4.
- When $i = 4$, there are 4 liters of water in tower 4. All 4 liters of water are turned into wine.

Hence, $W(a, b, c) = 1 + 4 + 2 + 4 = 11$ after the first update.
The second update modifies the arrays to $a = [3, 5, 3, 3]$, $b = [1, 1, 2, 8]$, and $c = [5, 1, 1]$.

- When $i = 1$, there are 3 liters of water in tower 1 and 1 liter of water is turned into wine. The remaining 2 liters of water flow into tower 2.
- When $i = 2$, there are 7 liters of water in tower 2 and 1 liter of water is turned into wine. Even though there are 6 liters of water remaining, only 1 liter of water can flow to tower 3.
- When $i = 3$, there are 4 liters of water in tower 3 and 2 liters of water is turned into wine. Even though there are 2 liters of water remaining, only 1 liter of water can flow into tower 4.
- When $i = 4$, there are 4 liters of water in tower 4. All 4 liters of water are turned into wine.

Hence, $W(a, b, c) = 1 + 1 + 2 + 4 = 8$ after the second update.
The third update modifies the arrays to $a = [3, 5, 0, 3]$, $b = [1, 1, 0, 8]$, and $c = [5, 1, 0]$.

- When $i = 1$, there are 3 liters of water in tower 1 and 1 liter of water is turned into wine. The remaining 2 liters of water flow into tower 2.

- When $i = 2$, there are 7 liters of water in tower 2 and 1 liter of water is turned into wine. Even though there are 6 liters of water remaining, only 1 liter of water can flow to tower 3.
- When $i = 3$, there is 1 liter of water in tower 3 and 0 liters of water is turned into wine. Even though there is 1 liter of water remaining, no water can flow to tower 4.
- When $i = 4$, there are 3 liters of water in tower 4. All 3 liters of water are turned into wine.

Hence, $W(a, b, c) = 1 + 1 + 0 + 3 = 5$ after the third update.

Source

Codeforces Hello 2024

Editor

The development of a text editor is a hard problem. You need to implement an extra module for brackets coloring in text.

Your editor consists of a line with infinite length and cursor, which points to the current character. Please note that it points to only one of the characters (and not between a pair of characters). Thus, it points to an index character. The user can move the cursor left or right one position. If the cursor is already at the first (leftmost) position, then it does not move left.

Initially, the cursor is in the first (leftmost) character.

Also, the user can write a letter or brackets (either `(`, or `)`) to the position that the cursor is currently pointing at. A new character always overwrites the old value at that position.

Your editor must check, whether the current line is the `correct text`. Text is correct if the brackets in them form the *correct bracket sequence*.

Formally, correct text (CT) must satisfy the following rules:

- any line without brackets is CT (the line can contain whitespaces);
- If the first character of the string — is `(`, the last — is `)`, and all the rest form a CT, then the whole line is a CT;
- two consecutively written CT is also CT.

Examples of correct texts: `hello(codeforces)`, `round`, `((i)(write))edi(tor)s`, `( me)`. Examples of incorrect texts: `hello)oops(`, `round)`, `((me)`.

The user uses special commands to work with your editor. Each command has its symbol, which must be written to execute this command.

The correspondence of commands and characters is as follows:

- `L` — move the cursor one character to the left (remains in place if it already points to the first character);
- `R` — move the cursor one character to the right;
- any lowercase Latin letter or bracket (`(` or `)`) — write the entered character to the position where the cursor is now.

For a complete understanding, take a look at the first example and its illustrations in the note below.

You are given a string containing the characters that the user entered. For the brackets coloring module's work, after each command you need to:

- check if the current text in the editor is a correct text;
- if it is, print the least number of colors that required, to color all brackets.

If two pairs of brackets are nested (the first in the second or vice versa), then these pairs of brackets should be painted in *different* colors. If two pairs of brackets are not nested, then they can be painted in different or the same colors. For example, for the bracket sequence `()(())()()` the least number of colors is 2, and for the bracket sequence `(()(()())())(())` — is 3.

Write a program that prints the minimal number of colors after processing each command.

Input

The first line contains an integer n ($1 \le n \le 10^6$) — the number of commands.

The second line contains s — a sequence of commands. The string s consists of n characters. It is guaranteed that all characters in a string are valid commands.

Output

In a single line print *n* integers, where the *i*-th number is:

- -1 if the line received after processing the first *i* commands is not valid text,
- the minimal number of colors in the case of the correct text.

Examples

Note

In the first example, the text in the editor will take the following form:

1. `(\\^`
2. `(\\ ^`
3. `(a\\ ^`
4. `(a\\  ^`
5. `(ab\\  ^`
6. `(ab\\   ^`
7. `(ab)\\   ^`
8. `(ab)\\  ^`
9. `(a))\\  ^`
10. `(a))\\ ^`
11. `(())\\ ^`

Source

Codeforces Round 603 (Div. 2)

Paimon Segment Tree

Paimon just learns the persistent segment tree and decides to practice immediately. Therefore, Lumine gives her an easy problem to start:

Given a sequence $a_1, a_2, \cdots, a_n$ of length n, Lumine will apply m modifications to the sequence. In the i-th modification, indicated by three integers l_i, r_i $(1 \le l_i \le r_i \le n)$ and x_i, Lumine will change a_k to $(a_k + x_i)$ for all $l_i \le k \le r_i$.

Let $a_{i,t}$ be the value of a_i just after the t-th operation. This way we can keep track of all historial versions of a_i. Note that $a_{i,t}$ might be the same as $a_{i,t-1}$ if it hasn't been modified in the t-th modification. For completeness we also define $a_{i,0}$ as the initial value of a_i.

After all modifications have been applied, Lumine will give Paimon q queries about the sum of squares among the historical values. The k-th query is indicated by four integers l_k, r_k, x_k and y_k and requires Paimon to calculate

$$\sum_{i=l_k}^{r_k} \sum_{j=x_k}^{y_k} a_{i,j}^2$$

Please help Paimon compute the result for all queries. As the answer might be very large, please output the answer modulo $10^9 + 7$.

Input

There is only one test case in each test file.

The first line of the input contains three integers n, m and q $(1 \le n, m, q \le 5 \times 10^4)$ indicating the length of the sequence, the number of modifications and the number of queries.

The second line contains n integers $a_1, a_2, \cdots, a_n$ $(|a_i| < 10^9 + 7)$ indicating the initial sequence.

For the following m lines, the i-th line contains three integers l_i, r_i and x_i $(1 \le l_i \le r_i \le n, |x_i| < 10^9 + 7)$ indicating the i-th modification.

For the following q lines, the i-th line contains four integers l_i, r_i, x_i and y_i $(1 \le l_i \le r_i \le n, 0 \le x_i \le y_i \le m)$ indicating the i-th query.

Output

For each query output one line containing one integer indicating the answer modulo $10^9 + 7$.

Examples

Input

```
3 1 1
8 1 6
2 3 2
2 2 0 0
```

Output

```
1
```

Input

```
4 3 3
2 3 2 2
1 1 6
1 3 3
1 3 6
```

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

Output

```
180
825
8
```

Source

The 2021 ICPC Asia Nanjing Regional Contest (XXII Open Cup, Grand Prix of Nanjing)