UCPC 2026

Union of Clubs for Programming Contests
Summer Contest 2026

Finals

Official Problemset

July 12th, 2026, 11:00 – 16:00

Hosted and Organized by

Union of Clubs for Programming Contests

Sponsored by

김동현(kdh9949) 김영현(kipa00) 나정휘(jhnah917) 정재호(cubic)

UCPC 2026

Finals

Problem List

Please verify that the problem set contains a total of 14 problems.

A	!!New Game Start!!
B	11-Hating Numbers
C	C4ESAR
D	Emergency Supply Distribution
E	Shift
F	Multiples
G	Vector Rails
H	Pandastic
I	Yet Another Binary Problem
J	Infinite String
K	Gapcheon Bike Path
L	Morning Medicine
M	Exorcising the Evil Spirit
N	Zigzag

All problems have the same memory limit of 1024 MB.

Problem A. !!New Game Start!!

Time Limit 4 seconds Memory Limit 1024 megabytes

Nina has started playing a recently released game, Bitaro's Dungeon. The dungeon has floors ranging from floor $-10^{10^{10}}$ to floor $10^{10^{10}}$, and the player starts on floor 0.

The game consists of a total of N stages. Whenever Nina moves to the next stage, she moves between floors before beginning the battle. When moving from stage $i-1$ to stage i, she moves up one floor if $A_i = +1$, and moves down one floor if $A_i = -1$.

Before starting the game, Nina may purchase any number of the M available coupons. If she purchases coupon i for Y_i won, the following effect is applied for every t satisfying $L_i \le t \le R_i$:

- After entering stage t, if her current floor is lower than X_i, she is immediately moved to floor X_i.

Multiple coupons may apply to the same stage t. In this case, only the applicable coupon with the largest value of X_i takes effect.

This game also has a **checkpoint** feature. If Nina is currently on floor x, she may pay D won to set a checkpoint there. From then on, whenever she attempts to move down from floor x to floor $x-1$, she remains on floor x instead. Nina may set checkpoints multiple times. Note that she may set a checkpoint on floor 0 immediately after starting the game.

Nina has a total of W won available. Find the maximum possible floor Nina can be on after completing all stages, under the condition that the total amount spent on purchasing coupons and setting checkpoints does not exceed W won.

Input

The first line contains four integers N, M, W, D, separated by spaces. They denote the number of stages, the number of coupons, the amount of money available, and the cost of setting a checkpoint, respectively. ($1 \le N \le 500\,000$; $0 \le M \le 500\,000$; $1 \le W \le 10^{15}$; $1 \le D \le \min(10^9, W)$)

The second line contains N integers $A_1, A_2, \cdots, A_N$, separated by spaces. ($A_i \in \{-1, 1\}$)

The next M lines describe the coupons.

The i-th of these lines contains four integers L_i, R_i, X_i, Y_i, separated by spaces. They denote the first stage to which the coupon applies, the last stage to which the coupon applies, the minimum guaranteed floor, and the price of the coupon, respectively. ($1 \le L_i \le R_i \le N$; $0 \le X_i \le N$; $1 \le Y_i \le \min(10^9, W)$)

Output

Print the maximum possible floor Nina can be on after completing all stages, under the condition that the total amount spent on purchasing coupons and setting checkpoints does not exceed W won.

Example

standard input	standard output
14 3 4 2 1 1 -1 -1 1 1 1 -1 -1 -1 1 1 -1 -1 1 14 1 4 2 10 2 4 8 14 3 4	5
8 3 5 2 -1 -1 1 -1 -1 1 1 -1 1 1 2 3 1 8 1 5 6 8 2 5	3

Note

In the first example, it is optimal not to purchase any coupons and to set two checkpoints. Nina can set a checkpoint on floor 3 after completing stage 7, and another checkpoint on floor 5 after completing stage 12.

In the second example, it is optimal to purchase coupon 1 and set one checkpoint. Coupon 1 costs 3 won, and setting a checkpoint costs 2 won, so Nina spends a total of 5 won.

After purchasing coupon 1, Nina can set a checkpoint on floor 2 immediately after completing stage 1.

Stage	Floor
1	2
2	2
3	3
4	2
5	2
6	3
7	4
8	3

Problem B. 11-Hating Numbers

Time Limit 1 second Memory Limit 1024 megabytes

A positive integer n is called an **11-hating number** if it cannot be made divisible by 11 by changing at most one of its digits to a different digit between 0 and 9, inclusive. Changing the first digit to 0 is also allowed, and 0 is considered a multiple of 11.

You are given two positive integers A and B. Find the number of 11-hating numbers between A and B, inclusive.

Input

The first line contains the integer A.

The second line contains the integer B.

$(1 \le A \le B \le 10^{200\,000})$

Output

Print the number of 11-hating numbers between A and B, inclusive.

Example

standard input	standard output
1 314159	2

Note

147 is not an 11-hating number because it can be made divisible by 11 by changing the digit 7 to 3.

545 is an 11-hating number.

Problem C. C4ESAR

Time Limit 1 second Memory Limit 1024 megabytes

You are given a string S consisting only of uppercase English letters and digits. You may apply the following operations to S any number of times:

- Apply the Caesar cipher to S.
- Delete one occurrence of `C4` that is a substring of S. If there are multiple such occurrences, you may delete any one of them.

The Caesar cipher replaces each uppercase English letter and each digit in a string according to the rules `A`→`B`→`C`→⋯→`Z`→`A` and `0`→`1`→`2`→⋯→`9`→`0`, respectively. For example, applying the Caesar cipher to `UCPC2026`, `C4ESAR`, and `ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789` results in `VDQD3137`, `D5FTBS`, and `BCDEFGHIJKLMNOPQRSTUVWXYZA1234567890`, respectively.

A substring is a string that appears as a contiguous segment of another string. For example, `U`, `PC`, and `UCPC` are substrings of `UCPC`, whereas `C4ESAR`, `CU`, and `CC` are not.

Determine whether it is possible to make S empty by applying the operations appropriately. If it is possible, find the minimum total number of operations required.

Input

The first line contains a string S consisting only of uppercase English letters and digits. ($1 \le |S| \le 10^6$)

Output

Print the minimum total number of operations required to make S empty. If it is impossible, print `-1` instead.

Example

standard input	standard output
`B3BC43`	`4`
`UCPC2026`	`-1`

Note

In the first example, S can be made empty by applying the operations in the following order:

1. Delete the fourth and fifth characters of S. Then, S becomes `B3B3`.
2. Apply the Caesar cipher to S. Then, S becomes `C4C4`.
3. Delete the third and fourth characters of S. Then, S becomes `C4`.
4. Delete the first and second characters of S. Then, S becomes empty.

In the second example, it is impossible to make S empty regardless of how the operations are applied.

Problem D. Emergency Supply Distribution

Time Limit 1 second Memory Limit 1024 megabytes

A sudden war has plunged the country of UCPC into crisis. With the entire domestic air transportation network paralyzed, the government has decided to distribute emergency supplies to each city using roads, the only remaining means of transportation.

The country of UCPC consists of N cities and $N-1$ roads. Each road connects two cities bidirectionally, and it is possible to travel between any pair of cities using only the roads.

The government currently possesses a total of N emergency supply items, where the k-th item has a value of k. The distribution operation starts in city 1, the capital. To compensate cities that receive their supplies later, the k-th city visited for the first time receives the k-th supply item. (City 1 is considered the first city visited and immediately receives supply item 1.)

The movement rules for the distribution operation are as follows.

- At each step, move from the current city to one of the adjacent cities connected to it by a road.
- If there is at least one adjacent city that has never been visited, choose one of them uniformly at random and move to it.
- If all adjacent cities have already been visited, return to the city from which the current city was reached when it was first visited.
- As soon as supplies have been distributed to all cities, terminate the operation.

When exactly one supply item is distributed to every city, including the capital, according to the rules above, calculate the expected value of the supply item received by each city.

Input

The first line contains the number of cities N. ($1 \le N \le 200\,000$)

Each of the next $N-1$ lines contains two integers a and b, separated by a space. ($1 \le a, b \le N;\ a \ne b$)

This means that cities a and b are connected by a road.

It is guaranteed that there is a path between every pair of cities.

Output

Print the answers in N lines.

For each $1 \le k \le N$, the k-th line should contain the expected value of the supply item received by city k.

It can be shown that each expected value is always rational. Suppose its irreducible fraction representation is $\frac{p}{q}$. You should print the integer M satisfying

$$q \cdot M \equiv p \pmod{998\,244\,353}.$$

($0 \le M < 998\,244\,353$)

It can be shown that such an integer exists and is unique.

Example

standard input	standard output
6 6 1 3 6 2 5 1 5 2 4	1 4 499122181 5 3 499122180
8 2 6 8 5 7 5 4 5 1 6 7 6 6 3	1 499122182 499122182 499122183 5 2 4 499122183

Note

In the first example, each of the two possible second cities is chosen with probability $\frac{1}{2}$. In each case, the order in which the cities receive their supplies is uniquely determined as follows.

- $1 \to 6 \to 3 \to 5 \to 2 \to 4$
- $1 \to 5 \to 2 \to 4 \to 6 \to 3$

The value of the supply item received by each city in the two orders is shown below.

City	Value in Order 1	Value in Order 2	Expected Value
1	1	1	$\frac{1}{2} \times 1 + \frac{1}{2} \times 1 = 1$
2	5	3	$\frac{1}{2} \times 5 + \frac{1}{2} \times 3 = 4$
3	3	6	$\frac{1}{2} \times 3 + \frac{1}{2} \times 6 = \frac{9}{2}$
4	6	4	$\frac{1}{2} \times 6 + \frac{1}{2} \times 4 = 5$
5	4	2	$\frac{1}{2} \times 4 + \frac{1}{2} \times 2 = 3$
6	2	5	$\frac{1}{2} \times 2 + \frac{1}{2} \times 5 = \frac{7}{2}$

Problem E. Shift

Time Limit 1 second Memory Limit 1024 megabytes

Consider an $N \times N$ grid whose cells contain pairwise distinct values. We define two operations, RowShift(r) and ColShift(c), as follows:

- RowShift(r): Shift every element in row r one cell to the right, with the rightmost element moving to the leftmost cell.
- ColShift(c): Shift every element in column c one cell downward, with the bottommost element moving to the topmost cell.

Given a permutation $(P_1, P_2, \ldots, P_N)$ of $(1, 2, \ldots, N)$, performing the following $2N$ operations in order is called one **procedure**:

1st:	RowShift(1)
2nd:	ColShift(P_1)
3rd:	RowShift(2)
4th:	ColShift(P_2)
	⋮
$(2N-1)$st:	RowShift(N)
$2N$th:	ColShift(P_N)

After repeating the **procedure** sufficiently many times, the grid is guaranteed to return to its initial state. Find the minimum number of repetitions required. Since the answer can be very large, print it modulo 998 244 353.

Input

The first line contains an integer N. ($1 \le N \le 200\,000$)

The second line contains N integers $P_1, P_2, \ldots, P_N$, separated by spaces. $(P_1, P_2, \ldots, P_N)$ is a permutation of $(1, 2, \ldots, N)$.

Output

Print the minimum number of times the **procedure** must be repeated, modulo 998 244 353.

Example

standard input	standard output
2 1 2	3

standard input	standard output
3 1 3 2	3

Problem F. Multiples

Time Limit 1 second Memory Limit 1024 megabytes

You are given positive integers N and M, along with M pairs $(P_1, Q_1), (P_2, Q_2), \ldots, (P_M, Q_M)$, where $P_i < Q_i$.

A sequence of positive integers $A_1, A_2, \ldots, A_N$ is called a **good sequence** if it satisfies both of the following conditions.

1. For every $1 \le i \le N$, the indices of the elements among $A_1, A_2, \ldots, A_N$ that are multiples of A_i form a single contiguous interval.
2. For every $1 \le i \le M$, A_{P_i} is not a multiple of A_{Q_i}.

The **beauty** of a **good sequence** is defined as the number of ordered pairs (i, j) $(1 \le i, j \le N)$ such that A_i is a multiple of A_j.

Determine whether a **good sequence** exists, and if it does, find the maximum possible **beauty**.

Input

The first line contains two integers N and M, separated by a space. $(2 \le N \le 100; 1 \le M \le 100)$

Each of the next M lines contains two integers P_i and Q_i, separated by a space. $(1 \le P_i < Q_i \le N)$

Output

If no **good sequence** exists, print `-1`.

Otherwise, print the maximum possible **beauty**.

Example

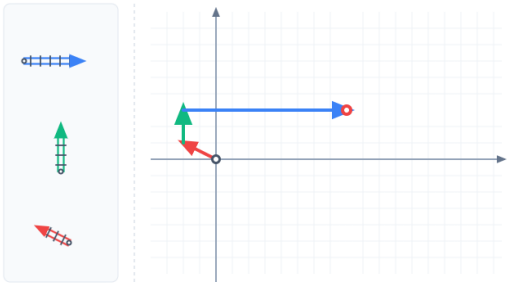
standard input	standard output
2 1 1 2	3

standard input	standard output
4 1 2 3	13

standard input	standard output
4 2 1 3 2 4	12

Problem G. Vector Rails

Time Limit 1 second Memory Limit 1024 megabytes



Recently, Mingyu has become engrossed in a railway construction game called 'Vector Rails'. In this game, starting from the origin $(0,0)$, he must construct a railway to each of several target cities located throughout the plane at minimum cost.

There are N construction modes available. If one unit of cost is spent using construction mode i, the railway is extended in the two-dimensional plane by the direction and length of the vector $\vec{v}_i = (a_i, b_i)$. This vector represents both the "direction of travel" and the "distance traveled per unit cost" of the corresponding construction mode.

Railway construction follows these rules:

- Railway construction always starts at the origin $(0,0)$.
- You may choose any construction mode i and spend any desired amount of cost t to extend the railway. Here, t may be any nonnegative real number, and the railway is extended by the vector $t\vec{v}_i$.
- At any point during construction, you may switch to another construction mode and continue building. There is no restriction on the number or order of mode changes.

Given the coordinates of Q target cities, write a program that independently computes, for each target city, the minimum construction cost required to connect it to the origin.

Input

The first line contains two integers N and Q, the number of construction modes and the number of target cities, respectively. $(1 \le N, Q \le 200\,000)$

Each of the next N lines contains two integers a_i and b_i, which form the vector $\vec{v}_i = (a_i, b_i)$ of construction mode i. $(-10^9 \le a_i, b_i \le 10^9;\ (a_i, b_i) \neq (0,0))$

Each of the following Q lines contains two integers p_x and p_y, the coordinates of a target city. $(-10^9 \le p_x, p_y \le 10^9)$

The vectors of different construction modes may be identical, and a target city may be located at the origin $(0,0)$.

Output

Print Q lines. For each target city, print the minimum construction cost required to reach it.

If the city cannot be reached using any combination of construction modes, print `-1` instead.

If the city is reachable, print the answer as a real number. Your answer is considered correct if its absolute or relative error does not exceed 10^{-9}.

Example

standard input	standard output
3 4 4 0 0 3 -2 1 2 3 -4 2 2 -1 0 0	1.5 2 -1 0
4 3 1 1 -1 1 -1 -1 1 -1 0 5 -3 -1 2 -3	5 3 3

Note

In the first example, the vectors of the three construction modes are $\vec{v}_1 = (4,0)$, $\vec{v}_2 = (0,3)$, and $\vec{v}_3 = (-2,1)$.

- $(2,3)$: Spend 0.5 units of cost using construction mode 1 to reach $(2,0)$, then switch to construction mode 2 and spend 1 unit of cost to reach $(2,3)$. The total construction cost is 1.5.
- $(-4,2)$: Spend 2 units of cost using construction mode 3 to reach $(-4,2)$ directly. No switch to another construction mode is necessary, and the total construction cost is 2.
- $(2,-1)$: Since the y-component of every construction mode is nonnegative, there is no way to extend the railway southward. Therefore, this city is unreachable.
- $(0,0)$: Since this is the origin itself, the total construction cost is 0.

In the second example, the four construction-mode vectors point northeast, northwest, southwest, and southeast, allowing construction in every direction.

- $(0,5)$: Spend 2.5 units of cost using construction mode 1 to reach $(2.5,2.5)$, then spend 2.5 units of cost using construction mode 2 to reach $(0,5)$. The total construction cost is 5.
- $(-3,-1)$: Spend 1 unit of cost using construction mode 2 to reach $(-1,1)$, then spend 2 units of cost using construction mode 3 to reach $(-3,-1)$. The total construction cost is 3.
- $(2,-3)$: Spend 2.5 units of cost using construction mode 4 to reach $(2.5,-2.5)$, then spend 0.5 units of cost using construction mode 3 to reach $(2,-3)$. The total construction cost is 3.

Problem H. Pandastic

Time Limit 1 second Memory Limit 1024 megabytes

There are N pandas living in the UCPC Zoo. Each panda likes at least one of the following three kinds of sticks: lipsticks, cheese sticks, and joysticks. For the pandas, the animal welfare organization UCPC animals has brought a bundle containing a total of N sticks: A lipsticks, B cheese sticks, and C joysticks. They will distribute the sticks according to the following rules.

1. Arrange the pandas in a line in any desired order.
2. The sticks are stacked inside the bundle. Starting from the top, take out one stick at a time and give it to the panda at the front of the line.
3. If the panda likes the stick it receives, it keeps the stick and leaves the line. Otherwise, it passes the stick to the panda behind it. This process continues until some panda leaves the line or the panda at the end of the line receives the stick.
4. If the panda at the end of the line receives a stick it does not like, the distribution event immediately ends in failure.

UCPC animals does not know the order in which the sticks are stacked inside the bundle. Determine whether there exists an ordering of the pandas such that all sticks can be distributed successfully regardless of the order in which the sticks are taken out. If such an ordering exists, find any one of them and help UCPC animals complete the event successfully!

Input

The first line contains the number of test cases T. $(1 \le T \le 10\,000)$

Each test case is given in the following format.

The first line contains the number of pandas N. $(1 \le N \le 500\,000)$

The second line contains three integers A, B, and C, separated by spaces, representing the numbers of the three kinds of sticks. $(0 \le A, B, C \le N;\ A + B + C = N)$

The next N lines contain information about the pandas. The i-th of these lines contains an integer K_i, the number of kinds of sticks liked by panda i, followed by K_i characters separated by spaces. $(1 \le K_i \le 3)$

Each character is one of `a`, `b`, and `c`, representing lipsticks, cheese sticks, and joysticks, respectively. No character appears more than once in the information for a single panda.

The sum of N over all test cases does not exceed 500 000.

Output

For each test case, print `Yes` on one line if there exists an ordering of the pandas such that all sticks can be distributed successfully regardless of the order in which the sticks are taken out. Otherwise, print `No`.

If such an ordering exists, print N distinct integers between 1 and N, separated by spaces, on the next line, representing the pandas in order from the front of the line to the back. If multiple orderings are possible, print any one of them.

Example

standard input	standard output
2 3 1 1 1 2 a b 3 c a b 1 b 3 3 0 0 2 a b 3 c a b 1 b	Yes 3 1 2 No

Note

In the first test case, arranging the pandas in the order 3, 1, 2 allows all sticks to be distributed successfully regardless of the order in which they are taken out.

In the second test case, the bundle contains no cheese sticks, so there is no way for panda 3 to receive a stick it likes.

Problem I. Yet Another Binary Problem

Time Limit 5 seconds Memory Limit 1024 megabytes

You are given an $N \times M$ grid A consisting only of 0s and 1s. You may perform either of the following two types of operations any number of times:

1. Remove one 0 and one 1 from the same row.
2. Remove one 0 and one 1 from the same column.

A cell whose number has been removed becomes empty. Find one way to maximize the number of operations performed.

Input

The first line contains two integers N and M, separated by a space. $(1 \le N, M \le 5000)$

Over the next N lines, the $(i+1)$-st line contains M digits $A_{i1}, A_{i2}, \cdots, A_{iM}$ without spaces, representing the grid A. Here, A_{ij} denotes the number written in row i, column j of the grid. $(0 \le A_{ij} \le 1)$

Output

Print N lines, each containing M characters $B_{i1}, B_{i2}, \cdots, B_{iM}$ without spaces, indicating by which type of operation each number was removed.

If the number in row i, column j was removed by an operation of type 1, let $B_{ij} = 1$. If it was removed by an operation of type 2, let $B_{ij} = 2$. If it was not removed, let $B_{ij} = x$.

The output must represent a way to maximize the total number of operations.

Example

standard input	standard output
3 4 0111 1001 1000	1212 1111 1212
2 2 00 10	2x 2x

Problem J. Infinite String

Time Limit 6 seconds Memory Limit 1024 megabytes

You are given a string S and Q intervals. The intervals are numbered from 1 to Q in the order they are given in the input, and the i-th interval is represented by two integers l_i and r_i.

The characters of S are numbered from 1 to $|S|$ from left to right. For $1 \le l \le r \le |S|$, let $S[l, r]$ denote the substring formed by concatenating the l-th through the r-th characters of S in order. For each i, let $X_i = S[l_i, r_i]$.

The infinite string X^∞ of a string X is the infinite-length string obtained by concatenating infinitely many copies of X. In other words, $X^\infty = X \cdot X \cdot X \cdots$.

Sort $X_1^\infty, X_2^\infty, \ldots, X_Q^\infty$ in ascending lexicographical order and output the resulting order.

When two infinite strings are identical, the interval with the smaller index must come first. In other words, if $X_i^\infty = X_j^\infty$ and $i < j$, then i must precede j.

Input

The first line contains a string S consisting only of lowercase English letters. $(2 \le |S| \le 150\,000)$

The second line contains the number of intervals Q. $(1 \le Q \le 1\,000\,000)$

Each of the next Q lines contains two integers l_i and r_i, separated by a space, representing the i-th interval. $(1 \le l_i \le r_i \le |S|)$

Output

Print the indices of the intervals in sorted order, separated by spaces.

The intervals are numbered $1, 2, \ldots, Q$ in the order they are given in the input.

Example

standard input	standard output
abacaba 4 1 2 1 3 5 6 2 4	2 1 3 4

standard input	standard output
ababababca 10 1 4 1 6 2 5 9 10 1 1 2 2 8 9 7 10 3 9 10 10	5 10 1 2 9 8 3 6 7 4

Note

In the first example, the given substrings are $X_1 = \texttt{ab}$, $X_2 = \texttt{aba}$, $X_3 = \texttt{ab}$, and $X_4 = \texttt{bac}$.

The first six characters of their corresponding infinite strings are as follows.

- $X_1^\infty = \texttt{ababab}\cdots$
- $X_2^\infty = \texttt{abaaba}\cdots$
- $X_3^\infty = \texttt{ababab}\cdots$
- $X_4^\infty = \texttt{bacbac}\cdots$

Comparing them lexicographically gives $X_2^\infty < X_1^\infty = X_3^\infty < X_4^\infty$.

Therefore, the sorted order is $2, 1, 3, 4$.

Since X_1^∞ and X_3^∞ are identical, interval 1, which has the smaller index, comes before interval 3.

Problem K. Gapcheon Bike Path

Time Limit 1.5 seconds Memory Limit 1024 megabytes

Jaewon went for a walk along the Gapcheon River to refresh himself. A straight bike path runs along the riverbank. There are N bicycles traveling along the bike path, each at a constant velocity.

Let the time when Jaewon starts his walk be 0, and the time when he finishes his walk be T. The position of the i-th bicycle is s_i when Jaewon starts his walk and e_i when he finishes it. The values s_i are pairwise distinct, and the values e_i are also pairwise distinct.

During Jaewon's walk, each bicycle meets at most K other bicycles. Two bicycles are considered to meet when their positions become equal.

Jaewon wants to calculate the distance from an arbitrary position x to the nearest bicycle at an arbitrary time t. Help Jaewon answer Q queries.

Input

The first line contains three integers N, T, and K, separated by spaces. They denote the number of bicycles, the time when Jaewon finishes his walk, and the maximum number of other bicycles that each bicycle meets, respectively. ($1 \le N \le 1\,000\,000$; $1 \le T \le 10^9$; $1 \le K \le 1\,000$)

Each of the next N lines contains two integers s_i and e_i, the initial and final positions of the i-th bicycle, respectively. ($0 \le s_i, e_i \le 10^9$) The values s_i are pairwise distinct, and the values e_i are also pairwise distinct.

The next line contains an integer Q, the number of queries. ($1 \le Q \le 10\,000$)

Each of the next Q lines contains two integers t and x, representing a query asking for the distance from position x to the nearest bicycle at time t. ($0 \le t \le T$; $0 \le x \le 10^9$)

It is guaranteed that, in the given input, each bicycle meets at most K other bicycles.

Output

For each query, print the answer on a separate line as an irreducible fraction in the form p/q, where p is a nonnegative integer and q is a positive integer.

Example

standard input	standard output
6 12 2 40 100 0 50 110 180 10 0 130 160 70 20 6 0 33 6 45 12 170 3 57 8 181 5 44	7/1 0/1 10/1 1/2 73/3 31/6

Problem L. Morning Medicine

Time Limit 1 second Memory Limit 1024 megabytes

Subin received a prescription at a hospital and purchased enough medicine for N days from a pharmacy. Each day's medicine consists of one packet for the morning, one for lunch, and one for dinner. The resulting $3N$ medicine packets are connected in a row in the following order:

morning medicine, lunch medicine, dinner medicine, morning medicine, lunch medicine, dinner medicine, $\cdots$

An ordinary person could simply eat all three meals every day and take the medicine packets by cutting them off one by one from the front. However, Subin's daily routine is not ordinary. On some days, Subin eats only dinner, while on others, Subin skips breakfast. Therefore, rather than taking the medicine in order from the front, Subin wants to take it according to the order of their meals. Of course, Subin intends to take all the medicine without leaving any packet unused.

However, Subin strongly dislikes seeing the connected medicine packets split into two separate pieces. Therefore, whenever Subin takes medicine, they cut off exactly one packet from either the front or the back of the remaining connected packets. Cutting off a packet from the middle, which would split the remaining packets into two pieces, is not allowed.

Given the order in which Subin wants to take the medicine, determine whether all the medicine can be taken in that order while following the condition above.

Input

The first line contains an integer N, the number of days' worth of medicine purchased. ($1 \le N \le 1\,000\,000$)

The second line contains a string S of length $3N$, representing the order in which Subin wants to take the medicine. The string S consists only of `B`, `L`, and `D`, and each of the three characters appears exactly N times.

The characters `B`, `L`, and `D` represent morning medicine, lunch medicine, and dinner medicine, respectively.

Output

Print `Yes` if Subin can take all the medicine in the order given by S while following the condition above. Otherwise, print `No`.

Example

standard input	standard output
1 LBD	No
2 BLDDBL	Yes

Note

In the first example, the initial sequence of medicine packets is, from front to back, morning medicine, lunch medicine, and dinner medicine. The first medicine to be taken is the lunch medicine, but it is at neither the front nor the back, so it cannot be taken while satisfying the condition.

In the second example, Subin can take all the medicine in the given order by cutting off packets from the front, front, front, back, front, and front, in that order.

Problem M. Exorcising the Evil Spirit

Time Limit 2 seconds Memory Limit 1024 megabytes

An evil spirit has infiltrated a street in Shinjuku. The street is shaped like a line segment connecting an entrance at its left endpoint to an exit at its right endpoint, and the distance between the two endpoints is $10^9 + 1$.

There are N sorcerers on the street. They are numbered from 1 to N in increasing order of distance from the entrance. Sorcerer i is located X_i units to the right of the entrance, where X_i is a positive integer, and all sorcerers are at distinct positions. Each sorcerer is also assigned one of two colors: red or blue.

Each curse is specified by two integers a and b. The evil spirit curses sorcerers a and b, where $1 \le a < b \le N$ always holds. A cursed region is then created on the line segment connecting the two sorcerers; that is, the cursed region is the interval $[X_a, X_b]$.

Once the cursed region is created, the sorcerers choose a point within it as the **core position**. The coordinate of the core position may be a real number. Let the core position be c. The core position may coincide with the position of a sorcerer.

After the core position is chosen, the cursed region is divided into red and blue exorcism regions according to the following rules.

For each $a \le i < b$, consider the interval $[X_i, X_{i+1}]$ between two adjacent sorcerers.

- If $X_{i+1} \le c$, the entire interval $[X_i, X_{i+1}]$ becomes an exorcism region of the same color as sorcerer i.
- If $c \le X_i$, the entire interval $[X_i, X_{i+1}]$ becomes an exorcism region of the same color as sorcerer $i+1$.
- If $X_i < c < X_{i+1}$, the interval $[X_i, c]$ becomes an exorcism region of the same color as sorcerer i, while the interval $[c, X_{i+1}]$ becomes an exorcism region of the same color as sorcerer $i+1$.

In other words, relative to the core position c, the portion on the left follows the color of the sorcerer at the left endpoint, while the portion on the right follows the color of the sorcerer at the right endpoint.

Under these rules, every point in the cursed region is covered by an exorcism region, and distinct exorcism regions may meet only at their endpoints.

When the core position is c, let $R(c)$ be the total length of the red exorcism regions and $B(c)$ be the total length of the blue exorcism regions. The **disharmony** of the exorcism ritual is the absolute difference between these two values, namely $|R(c) - B(c)|$.

To increase the ritual's chance of success, the sorcerers want to choose the core position c appropriately so that the disharmony is minimized.

Given the information about the sorcerers and the curses cast by the evil spirit, write a program that finds the minimum possible disharmony for each curse.

Input

The first line contains two integers N and Q, the number of sorcerers and the number of curses, respectively. ($2 \le N \le 200\,000$; $1 \le Q \le 200\,000$)

Each of the next N lines contains the coordinate X_i and the color C_i of sorcerer i. ($1 \le X_1 < X_2 < \ldots < X_N \le 10^9$, $C_i =$ R or B)

Each of the following Q lines contains two integers a and b, describing a curse. They are the indices of the two sorcerers cursed by the evil spirit. ($1 \le a < b \le N$)

Output

For each curse, print the minimum possible disharmony on a separate line. It can be proven that the minimum disharmony is always an integer.

Example

standard input	standard output
4 2 1 R 6 B 8 B 12 R 1 4 2 3	0 2

Note

Curse 1: If the core position is chosen as $c = 2.5$, the intervals $[1, 2.5]$ and $[8, 12]$ become red exorcism regions, while the intervals $[2.5, 6]$ and $[6, 8]$ become blue exorcism regions. The total lengths of the red and blue exorcism regions are both 5.5, so the disharmony is 0. Choosing $c = 11.5$ also results in a disharmony of 0.

Curse 2: If the core position is chosen as $c = 8$, the entire interval $[6, 8]$ becomes a blue exorcism region. Since there is no red sorcerer in the cursed region, the disharmony cannot be made smaller than 2.

Problem N. Zigzag

Time Limit 1 second Memory Limit 1024 megabytes

A **zigzag string** is a binary string in which `0` and `1` alternate. For example, `0`, `101`, and `010101` are zigzag strings, whereas `00` and `0110` are not.

For a binary string A, define $f(A)$ as the minimum number of zigzag subsequences into which A can be partitioned. For example, `11000101` can be partitioned into the following three zigzag subsequences: `10`, `101`, and `010`. The first row represents the original string, and the following three rows represent the three subsequences.

A	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**1**
Zigzag string 1		1		0				
Zigzag string 2	1				0			1
Zigzag string 3			0			1	0	

It can also be shown that there is no way to partition `11000101` into at most two zigzag subsequences. Therefore, $f(\texttt{11000101}) = 3$.

Given a binary string A of length N, find the sum of $f(s)$ over all substrings s of A.

More formally, for every pair (l, r) satisfying $1 \le l \le r \le N$, let $A[l, r]$ denote the string $A_l A_{l+1} \dots A_r$, consisting of the l-th through the r-th characters of A. Compute

$$\sum_{l=1}^{N} \sum_{r=l}^{N} f(A[l, r]).$$

Input

The first line contains an integer N, the length of A. $(2 \le N \le 300\,000)$

The second line contains the binary string A of length N.

Output

Print the answer as an integer.

Example

standard input	standard output
4 1101	13
8 10000101	79