



2026 ICPC Bolivia National Contest - Bolivian Programming Competition, Phase 2

Task Sheet

2026

General information

This problem set consists of 24 pages (not counting the cover), numbered from 1 to 24. Check that the problem set is complete.

Input

- The input must be read from standard input.
- All lines of the input, including the last one, end with a newline character (`\n`).
- When the input contains several values separated by spaces, there is exactly one blank space between two consecutive values on the same line.

Output

- The output must be written to standard output.
- The output must follow the format specified in the problem statement. The output must not contain any additional data.
- All lines of the output, including the last one, must end with a newline character (`\n`).
- When an output line shows several values separated by spaces, there must be exactly one blank space between two consecutive values.
- When an output value is a real number, use at least the number of decimal places corresponding to the precision required in the problem statement.

Problem A. Borders

Time limit: 2000 ms

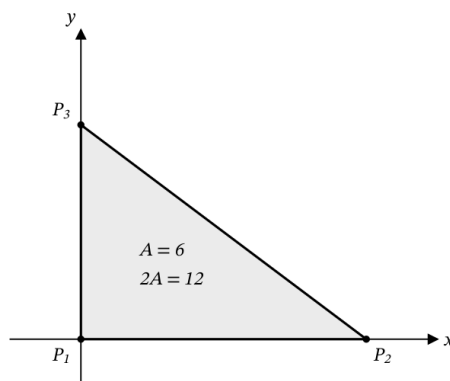
Memory limit: 256 MiB

In the country of Logoria there are N independent towns. Each town is located at a point of the plane with integer coordinates (x_i, y_i) .

At first, each town forms its own community. Then an era of conflicts begins. Each conflict involves two towns a and b : after it, the communities to which towns a and b belong become a single community (it does not matter who wins the dispute). If a and b already belong to the same community, the conflict produces no change.

The territory of a community is the smallest convex region that contains all of its towns. The boundary of this territory is known as the **convex hull** of its points.

After each conflict, determine **twice** the area of the territory of the resulting community. If all the towns of a community lie on the same line, its area is considered to be zero. It can be shown that twice the area is always an integer.



Three towns forming a triangle of area 6.

Input

The first line contains two integers N and M ($2 \leq N \leq 2000$, $1 \leq M \leq 100000$), where N is the number of towns and M is the number of conflicts.

Each of the following N lines contains two integers x_i and y_i ($-1000000 \leq x_i, y_i \leq 1000000$), the coordinates of the i -th town. The towns are numbered from 1 to N , and all of them have distinct coordinates: $(x_i, y_i) \neq (x_j, y_j)$ for $i \neq j$.

Each of the following M lines contains two integers a and b ($1 \leq a, b \leq N$), meaning that the communities containing towns a and b merge. A conflict may involve two towns that already belong to the same community.

Output

Print M lines. The j -th line must contain twice the area of the territory of the community that contains the towns of the j -th conflict, immediately after applying that conflict.

Examples

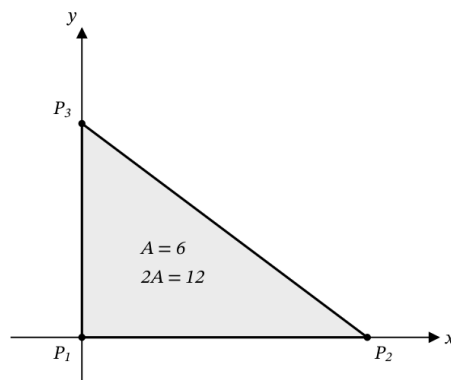
Sample input 1	Sample output 1
<pre> 5 4 0 0 4 0 0 3 4 3 2 1 1 2 2 3 4 5 1 4 </pre>	<pre> 0 12 0 24 </pre>

Explanation of sample 1

Initially there are five independent towns: $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$ and $\{5\}$.

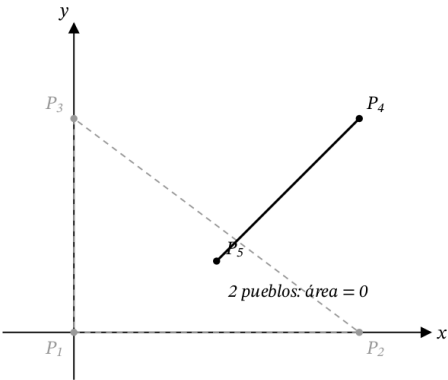
After the conflict between towns 1 and 2, the resulting community has only two towns, so the answer is 0.

The conflict between 2 and 3 joins $\{1, 2\}$ with $\{3\}$: the three towns form a triangle of area 6, so twice that value, 12, is printed.



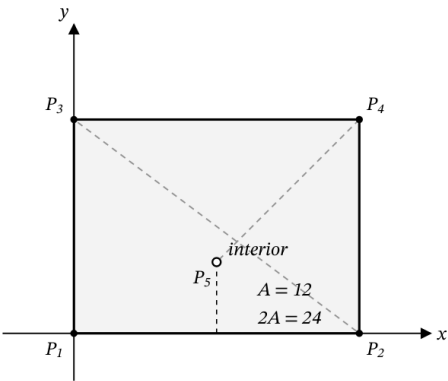
Community $\{1, 2, 3\}$ forms a triangle with twice the area equal to 12.

The conflict between 4 and 5 forms a community of two towns and again produces 0.



In the same plane, community $\{4,5\}$ is just a segment (2 towns, area 0); community $\{1,2,3\}$ is shown in gray for reference.

Finally, the conflict between 1 and 4 joins $\{1,2,3\}$ with $\{4,5\}$. Town $5 = (2,1)$ lies inside the rectangle formed by the other four towns and does not change the boundary; that rectangle has area 12, so the answer is 24.



When the two communities (dashed lines) merge, town 5 ends up in the interior of the new boundary (solid line).

Sample input 2	Sample output 2
4 4	0
0 0	0
1 0	0
2 0	4
0 2	
1 2	
1 2	
2 3	
1 4	

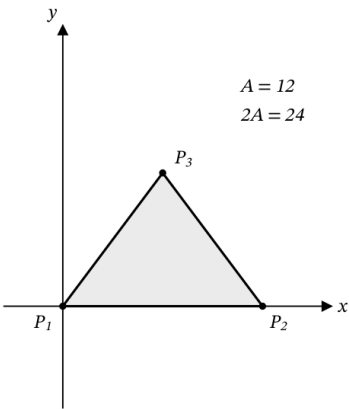
Explanation of sample 2

The first three towns lie on the same line, so as long as the community does not include town 4 the answer is 0; in particular, the second conflict (1-2) does not change the community, because 1 and 2 already belonged to $\{1,2\}$. When town 4 joins, the hull becomes the triangle with vertices $(0,0)$, $(2,0)$ and $(0,2)$, whose doubled area is 4.

Sample input 3	Sample output 3
6 5 0 0 6 0 3 4 2 -2 4 -2 3 7 1 2 1 3 4 5 4 6 1 4	0 24 0 18 58

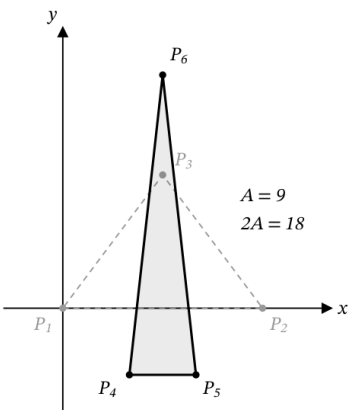
Explanation of sample 3

Conflicts 1-2 and 1-3 form community {1, 2, 3}, whose boundary is triangle $P_1P_2P_3$ (twice the area: 24).



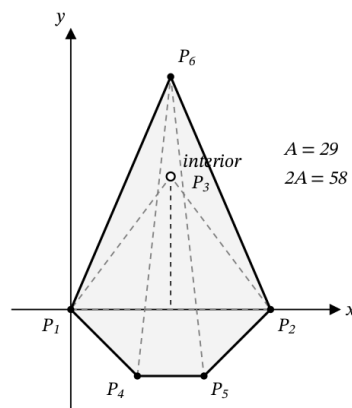
Community {1, 2, 3} forms a triangle with twice the area equal to 24.

Independently, conflicts 4-5 and 4-6 form {4, 5, 6}, with boundary $P_4P_5P_6$ (twice the area: 18).



In the same plane, community {4, 5, 6} forms a triangle with twice the area equal to 18 (community {1, 2, 3} is shown in gray for reference).

Conflict 1-4 joins both communities. The new boundary is pentagon $P_1P_4P_5P_2P_6$, and town 3 ends up in its interior despite having been a vertex of its own community's boundary just before the merge: this is why it is not enough to combine the two previous boundaries; the convex hull of the union must be recomputed. Twice the area of the pentagon is 58.



When the two communities (dashed lines) merge, town 3 stops being a vertex of the new boundary (solid line).

Notes

Problem B. MiniMax

Time limit: 1000 ms
Memory limit: 256 MiB

You are given an array of n integers, which may be negative, zero or positive, and a positive integer k .

You must answer q queries. Each query gives two positions l and r . Your task is to split the subarray $a[l..r]$ into the **minimum possible number of non-empty contiguous segments**, such that, in each segment, the difference between its maximum element and its minimum element is less than or equal to k .

The segments must cover exactly the queried subarray, without overlapping and keeping the original order of the elements. A segment with a single element has difference zero.

Positions are numbered starting from 1. The array and the value of k are the same for all queries. The queries are independent and do not modify the array.

Input

The first line contains two integers n and q ($1 \leq n, q \leq 10^5$): the length of the array and the number of queries.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($-10^9 \leq a_i \leq 10^9$).

The third line contains an integer k ($1 \leq k \leq 10^9$).

Each of the following q lines contains two integers l and r ($1 \leq l \leq r \leq n$), representing a query on $a[l..r]$.

The input contains a single test case and is read from standard input.

Output

For each query, print on a line of standard output the minimum number of segments required, in the same order in which the queries are received.

Examples

Sample input 1	Sample output 1
7 3	3
1 5 3 2 7 6 4	2
3	1
1 7	
2 5	
3 4	

Explanation of sample 1

For $[1, 7]$, an optimal partition is $[1] \mid [5, 3, 2] \mid [7, 6, 4]$. The differences are 0, 3 and 3. It is mandatory to cut between 1 and 5, and between 2 and 7, because each of those consecutive pairs has a difference greater than 3. Therefore, at least three segments are needed.

For $[2, 5]$, an optimal partition is $[5, 3, 2] \mid [7]$. The whole subarray has difference $7 - 2 = 5 > 3$, so a

single segment is not enough.

For $[3, 4]$, the segment $[3, 2]$ has difference 1, so the answer is 1.

Sample input 2**Sample output 2**

6 4	3
-3 -1 0 2 2 -2	1
3	1
1 6	2
2 5	
3 3	
5 6	

Explanation of sample 2

For $[1, 6]$, an optimal partition is $[-3, -1, 0] \mid [2, 2] \mid [-2]$. There must be a cut before the first 2, because it cannot share a segment with -3 , and another cut between the last 2 and -2 , whose difference is 4. Those cuts are different, so at least three segments are needed.

For $[2, 5]$, the subarray $[-1, 0, 2, 2]$ has difference 3 and fits in a single segment.

For $[3, 3]$, the subarray contains only the zero and requires one segment.

For $[5, 6]$, the values 2 and -2 have difference 4 and must be placed in separate segments.

Notes

Problem C. Firewood in Corani

Time limit: 1200 ms
Memory limit: 256 MiB

The 2880 algorithms club went to Corani and is now preparing firewood for the fireplace. The little wood logs are cylindrical and stacked in the shape of a triangular pyramid: there are N logs at the base, $N - 1$ in the next row, and so on until a single log at the top.

Pakai came up with the idea of playing by removing logs from the base and discovered an interesting problem. Each log in an upper row rests on the two logs immediately below it. If it loses **either of its two supports**, it falls. This can make other logs fall in a chain reaction. Fallen logs are removed from the pile without hitting or displacing the others.

Pakai will remove Q logs from the base, one by one, waiting for all the falls to finish after each removal. The positions of the base are numbered from 1 to N , from left to right, and keep their original numbering throughout the whole process (fallen or removed logs never return to the pile).

How many logs remain in the whole pyramid after each removal?

Input

The first line contains two integers N and Q ($1 \leq N \leq 10^9$; $1 \leq Q \leq \min(N, 2 \times 10^5)$).

Each of the following Q lines contains an integer p_i ($1 \leq p_i \leq N$): the position of the base log that Pakai removes in operation i . All positions p_i are pairwise distinct.

Output

Print Q lines. On line i , write the total number of logs that remain in the whole pyramid after operation i (that is, after removing p_i and all the falls that this removal causes).

Examples

Sample input 1	Sample output 1
5 3 3 1 5	6 4 2

Explanation of sample 1

Initially there are 15 logs. After removing the one at position 3, two intact stretches of two positions each remain (positions 1, 2 and 4, 5), which support $3 + 3 = 6$ logs. After removing the one at position 1, the left stretch is reduced to position 2 alone and $1 + 3 = 4$ logs remain. Finally, after removing the one at position 5, only the base logs at positions 2 and 4 survive, each one supporting only itself: $1 + 1 = 2$ logs.

Sample input 2	Sample output 2
6 2 1 6	15 10

Explanation of sample 2

This case illustrates the ends of the base. Initially there are 21 logs. After removing position 1 (the left end), a single stretch with positions $2, \dots, 6$ survives, which supports 15 logs. After then removing position 6 (the right end), the stretch $2, \dots, 5$ survives, which supports 10 logs.

Notes

Problem D. Cookies for the Feed

Time limit: 1000 ms

Memory limit: 256 MiB

For an Instagram *reel* of **icpc_umss**, the team wants to bake letter-shaped cookies and use them to spell several words. Each word is prepared in one **batch**: all of its letters are baked at the same time, using one mold per cookie.

Each mold has a fixed letter and can be reused, but it must cool down for one full batch after each use. For example, a mold used in the first batch cannot be used in the second one, but it can be used in the third.

The words must be prepared in the given order, with no additional pauses or empty batches. At the beginning, all molds are cold. What is the minimum total number of molds that must be bought before starting?

Input

The first line contains an integer N ($1 \leq N \leq 1000$), the number of words.

Each of the following N lines contains a word S_i of between 1 and 1000 uppercase letters of the English alphabet (A to Z). Repeated letters require different molds within the same batch.

Output

Print an integer: the minimum total number of molds needed.

Examples

Sample input 1	Sample output 1
3 ANA LIA ANA	6

Explanation of sample 1

Three A molds, one N, one L and one I are bought. The first batch uses two A molds and the N mold; the second uses the third A mold, the L mold and the I mold. By the third batch the molds from the first batch are already cold, so the two A molds and the N mold can be reused.

Sample input 2	Sample output 2
1 ICPC	4

Explanation of sample 2

Two C molds, one I and one P are needed: the word ICPC uses the letter C twice in the same batch, so two different molds of that letter are required.

Notes

Problem E. Waso Number

Time limit: 250 ms

Memory limit: 256 MiB

A non-negative integer is called **waso** if, when written in base 10 without leading zeros, its digits read from left to right form a non-decreasing sequence. For example, 0, 5, 19, 122 and 123444 are waso, while 10, 21 and 102 are not.

The waso numbers, sorted from smallest to largest, are:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, $\dots$, 19, 22, 23, $\dots$, 99, 111, $\dots$

Given an integer k , find the k -th smallest waso number (considering 0 as the first one).

Input

The input consists of a single line with an integer k ($1 \leq k \leq 10^{18}$).

Output

Print a single integer: the k -th waso number.

Note that the answer may have several hundred digits, so it does not fit in standard 64-bit integer types.

Examples

Sample input 1	Sample output 1
1	0

Explanation of sample 1

The first waso number is 0.

Sample input 2	Sample output 2
11	11

Sample input 3	Sample output 3
1000000	113334555668999

Notes

Problem F. Lights for the SanSiCup

Time limit: 1000 ms

Memory limit: 256 MiB

Every Christmas, while decorating the hall, the kids of ICPC UMSS organize the SanSiCup. On a wall there is a panel with b lights, numbered from 0 to $b - 1$, and N remote controls. At the beginning all lights are off.

Each control i has an associated fixed mask a_i : when pressed, it toggles exactly the lights j such that bit j of a_i is one (lights that are off turn on and lights that are on turn off). The controls can be pressed in any order and as many times as desired; pressing the same control twice cancels its effect. Consequently, a **configuration** of lights (the subset of lights that are on) is reachable if and only if it is equal to the bitwise XOR of the masks of some subset of the original controls (the empty subset produces the configuration with all lights off).

Since the controls often get lost among the decorations, the kids want to build some additional backup controls. The effect of each new control is chosen **before** any loss happens, is fixed at that moment and cannot be changed afterwards; that effect must be equal to the XOR of the masks of some subset of the *original* controls (not of other new controls). A new control with mask 0 is valid, although useless.

Then, up to two arbitrary controls —original or new, in any combination— may be lost. Two different controls are always considered physical and separate even if they have exactly the same mask: losing one of them does not affect the other. Once the new controls are built and the identity of the lost controls is known, the kids can freely choose, for each configuration they want to reproduce, which remaining controls to press (different subsets for different configurations are allowed).

The kids want that, no matter which up to two controls are lost, **every** configuration that was reachable with the N original controls (without losses and without new controls) remains reachable using only the controls that survived. What is the minimum number of additional controls they need to build to guarantee this?

Input

The first line contains two integers N and b ($1 \leq N \leq 100000$, $1 \leq b \leq 60$): the number of original controls and the number of lights on the panel.

The second line contains N integers $a_1, a_2, \dots, a_N$ ($0 \leq a_i < 2^b$), the masks of the original controls, in decimal. The lights are numbered from 0 to $b - 1$: control i toggles light j if and only if bit j of a_i is one.

Several original controls may have exactly the same mask (including mask 0, a control that does not change any light); each one counts as an independent physical unit that can be lost separately.

Output

Print a single integer: the minimum number of additional controls the kids must build.

Examples

Sample input 1	Sample output 1
5 3 4 2 1 6 5	1

Explanation of sample 1

It is enough to build one additional control with mask 3 (equivalent to pressing the controls with masks 2 and 1). With the six available controls, every configuration reachable with the five original ones remains reachable even if any two of them are lost.

For example, if the controls with masks 4 and 2 are lost, the effect of the first one can be reconstructed by pressing those with masks 5 and 1, and the effect of the second one by pressing those with masks 3 and 1.

Without building any additional control, losing the controls with masks 2 and 6 already makes it impossible to obtain configuration 2.

Sample input 2	Sample output 2
3 3 4 2 1	3

Explanation of sample 2

The three original controls are linearly independent (they form a basis) and none of them has a backup: losing any one of them already makes it impossible to reproduce its effect. An optimal choice is to build the controls with masks 6, 5 and 3.

Sample input 3	Sample output 3
3 1 1 1 1	0

Explanation of sample 3

Even if two of the three controls are lost, there is always one left that can turn the only light on or off. There is no need to build any additional control.

Notes

Remember that:

- The effect of each new control is fixed before it is known which controls will be lost, and it must be expressible as the XOR of masks of *original* controls (never of other new controls).
- “Losing up to two controls” includes losing zero, one or two controls, in any combination of original and new ones.
- Two controls with the same mask are still distinct units: losing one does not imply losing the other.

Problem G. Sergio's Happiest Office

Time limit: 2100 ms

Memory limit: 256 MiB

Sergio runs an office with m windows arranged in a row. Blind j currently leaves an opening of a_j centimeters and cannot be opened beyond h_j centimeters.

Sergio's colleagues, Miguel and Racsó, convinced him that the team must get enough natural light: according to the wellness program, reasonable sun exposure helps maintain healthy vitamin D levels and improves the mood in the office. There are n office workers in total.

Office worker i receives light from all windows from l_i to r_i , inclusive. Their light level is the sum of the openings of those blinds.

While Miguel and Racsó are at lunch, Sergio can only **raise** blinds. So that the change is not obvious, the building controller allows at most k centimeters of total movement. He must choose non-negative integers $x_1, \dots, x_m$ such that

$$x_1 + x_2 + \dots + x_m \leq k$$

and

$$a_j + x_j \leq h_j$$

for each blind j . After the adjustment, the opening of blind j will be $a_j + x_j$ centimeters. It is guaranteed that at least one adjustment satisfying all the constraints exists.

The management wants to maximize the light received by the least illuminated office worker. Print only that maximum value.

Input

The first line contains three integers m , n and k ($1 \leq m, n \leq 2 \times 10^5$, $0 \leq k \leq 10^{12}$).

The second line contains m integers $a_1, \dots, a_m$ ($0 \leq a_j \leq 10^9$).

The third line contains m integers $h_1, \dots, h_m$ ($a_j \leq h_j \leq 10^9$).

Each of the following n lines contains two integers l_i and r_i ($1 \leq l_i \leq r_i \leq m$). Different office workers may have the same interval of windows.

Output

Print a single integer: the maximum possible value of the minimum illumination among all office workers.

Examples

Sample input 1

```
4 3 5
1 0 2 1
3 2 2 4
1 2
2 3
4 4
```

Sample output 1

```
3
```

Explanation of sample 1

Sergio can raise the blinds by 0, 2, 0, 3 centimeters. The light levels of the three office workers become 3, 4, 4. Achieving at least 4 for all of them would require three centimeters on the first two windows and three on the fourth, a total of six centimeters; Sergio only has five.

Notes

Problem H. Racsó's Toy Train

Time limit: 2600 ms

Memory limit: 256 MiB

Racsó has a toy train that he loves. It has n numbered stations joined by m bidirectional track segments. Initially, the train can travel from any station to any other.

Two distinct stations are **well connected** if, after any track segment comes loose, the train can still travel between those two stations. Only one segment comes loose at a time.

Miguel wants to make the train more reliable. For each of q plans he can add two colored track segments: one between stations a and b , and another between stations c and d .

For each plan, count the unordered pairs of distinct stations that satisfy the following three conditions:

- They are well connected if both new segments are added.
- They are not well connected if only the segment between a and b is added.
- They are not well connected if only the segment between c and d is added.

In each scenario, the segment that comes loose can be any of those that exist in it, including the two new ones.

The plans are independent: each one starts from Racsó's original train and no new segment is kept for the next plan.

A new segment may join the same stations as an original track or as the other proposed segment. They are different physical tracks: if one comes loose, the others keep working. The four stations of a plan do not have to be distinct.

Input

The first line contains three integers n , m and q ($2 \leq n \leq 2 \times 10^5$, $n - 1 \leq m \leq 3 \times 10^5$, $1 \leq q \leq 2 \times 10^5$).

The next m lines contain two integers u and v ($1 \leq u, v \leq n$, $u \neq v$), the endpoints of an original segment. The train is connected and does not contain two original segments with the same endpoints.

The next q lines contain four integers a , b , c and d ($1 \leq a, b, c, d \leq n$, $a \neq b$, $c \neq d$), describing a plan with two new segments.

Output

Print q lines. On each line, print the number of pairs corresponding to that plan. Each pair is counted only once, regardless of the order of its stations.

Examples

Sample input 1	Sample output 1
4 3 5 1 2 2 3 3 4 1 3 2 4 1 2 2 3 1 2 3 4 1 4 2 3 1 3 3 4	1 1 0 0 2

Explanation of sample 1

In the first plan, the only pair that needs both new segments is $\{1, 4\}$. With only the first one, the train depends on segment 3-4; with only the second one, it depends on 1-2.

In the second plan, additional copies of segments 1-2 and 2-3 are added. The pair $\{1, 3\}$ becomes well connected only when both copies are added.

In the last plan, the pairs that satisfy the conditions are $\{1, 4\}$ and $\{2, 4\}$.

Sample input 2	Sample output 2
5 4 1 1 2 1 3 1 4 1 5 2 3 4 5	4

Explanation of sample 2

The pairs are $\{2, 4\}$, $\{2, 5\}$, $\{3, 4\}$ and $\{3, 5\}$. The alternative routes of the train are allowed to pass through the same station: being well connected refers to a loose segment, not to a closed station.

Notes

Problem I. Backup Ink

Time limit: 3000 ms

Memory limit: 256 MiB

A print shop represents a strip of paper by a string s of length n . Each position of the string corresponds to a cell of the strip.

To test a pattern t , which must be a non-empty substring of s , the print shop starts with all cells unpainted. Then it performs one ink application for **each occurrence** of t in s .

If an occurrence starts at position i , its application paints all the cells of the interval $[i, i + |t| - 1]$. Applications may overlap: a cell is considered painted as long as at least one application covers it.

The print shop wants to measure what happens if an application fails. To do so, it cancels **exactly one** of the applications and keeps all the others. The **damage** of that cancellation is the number of cells that were painted and stop being painted.

A pattern is **resilient** if the damage is at most k , **no matter which of its applications is cancelled**.

Compute how many **distinct non-empty** substrings of s are resilient.

The tests of different patterns and the different cancellations are considered independently. Cancelling an application does not modify the characters of s nor the positions of its occurrences. The pattern does not need to paint the whole strip.

Two substrings are considered equal if they have the same length and the same characters, even if they appear at different positions.

Input

The first line contains two integers n and k ($1 \leq n \leq 2 \times 10^5$, $0 \leq k \leq n$): the length of the string and the maximum allowed damage.

The second line contains a string s of length n , consisting only of lowercase letters of the English alphabet.

Output

Print an integer: the number of distinct non-empty substrings that are resilient.

The answer may not fit in a 32-bit integer.

Examples

Sample input 1	Sample output 1
5 1 aaaaa	4

Explanation of sample 1

The resilient patterns are **a**, **aa**, **aaa** and **aaaa**.

For example, **aaa** occurs in the intervals $[1, 3]$, $[2, 4]$ and $[3, 5]$. When the first application is cancelled,

only cell 1 stops being painted. When the middle application is cancelled, no cell stops being painted. When the last one is cancelled, only cell 5 stops being painted.

The pattern `aaaaa` is not resilient: it has a single occurrence and, when its application is cancelled, all five cells stop being painted.

Sample input 2**Sample output 2**

15 1 aaaaabaaabaaaaa	4
-------------------------	---

Explanation of sample 2

The resilient patterns are `a`, `b`, `aa` and `aaaa`.

The pattern `aaa` is not resilient: when the application corresponding to $[7, 9]$ is cancelled, the three cells of that interval stop being painted.

On the other hand, `aaaa` occurs in $[1, 4]$, $[2, 5]$, $[11, 14]$ and $[12, 15]$. Cancelling any of those applications causes a damage of one cell.

Sample input 3**Sample output 3**

3 3 abc	6
------------	---

Explanation of sample 3

All distinct substrings are resilient: `a`, `b`, `c`, `ab`, `bc` and `abc`. Each of them has a single occurrence, but cancelling its application causes a damage of at most 3.

Notes

Problem J. The Single-Cut Cross

Time limit: 1000 ms

Memory limit: 256 MiB

In a workshop there are n cables laid out on a plane. Cable i is the straight segment joining $A_i = (x_{i,1}, y_{i,1})$ with $B_i = (x_{i,2}, y_{i,2})$ and has a value w_i .

At the origin $O = (0, 0)$ there is a machine with two lasers. Each laser occupies an **infinite line** passing through the origin. The two lines are perpendicular and can rotate together, always keeping an angle of 90° between them.

You must choose an orientation of the machine and activate it only once. Both lasers are activated simultaneously and go through all the cables they meet, without stopping.

An orientation is **valid** if it satisfies the following conditions:

- Neither of the two lasers passes through an endpoint of any cable.
- No cable is intersected by both lasers.

In a valid orientation, a cable is considered cut if one of the lasers intersects its interior, that is, the segment without its endpoints. Each cut cable contributes its value w_i ; cables that are not cut contribute nothing.

Find the maximum total value that can be obtained with a valid orientation. If there is none, print -1 .

It is not necessary to cut all the cables, or even one. However, if both lasers intersect the same cable, **the whole orientation is invalid**: you cannot simply discard that cable from the score.

It is guaranteed that no cable contains the origin. Cables may cross, share endpoints or overlap; they are considered independent objects and their values are counted separately.

Input

The first line contains an integer t ($1 \leq t \leq 10^4$), the number of test cases.

For each test case:

The first line contains an integer n ($1 \leq n \leq 2 \times 10^5$), the number of cables.

Each of the following n lines contains five integers:

$$x_{i,1} \quad y_{i,1} \quad x_{i,2} \quad y_{i,2} \quad w_i$$

with the constraints

$$-10^9 \leq x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2} \leq 10^9,$$

$$1 \leq w_i \leq 10^9.$$

The endpoints of each cable are distinct and the segment joining them does not contain $(0, 0)$.

The sum of n over all test cases does not exceed 2×10^5 .

Output

For each test case, print an integer: the maximum total value of the cut cables with a valid orientation, or -1 if there is no valid orientation.

Examples

Sample input 1	Sample output 1
4	11
3	5
4 1 4 2 5	-1
1 4 2 4 7	0
3 1 1 3 4	
2	
-2 1 2 1 5	
3 2 2 3 100	
3	
-3 -2 3 -2 1	
3 -2 1 4 1	
1 4 -3 -2 1	
1	
1 1 2 2 9	

Explanation of sample 1

In the first case, the lines $y = \frac{5}{2}x$ and $y = -\frac{2}{5}x$ form a valid orientation. The second and third cables are cut, obtaining $7 + 4 = 11$. There is no valid orientation that achieves a larger value.

In the second case, the coordinate axes form a valid orientation that cuts only the first cable, obtaining 5. Any orientation that cuts the cable of value 100 intersects the first cable with both lasers, so it is invalid.

In the third case, the cables form the outline of a triangle that contains the origin in its interior. Every line through the origin that avoids the vertices crosses two sides. Between the two lasers there are four intersections with only three cables, so at least one of them receives two cuts. There is no valid orientation.

In the fourth case, the cable lies on the same ray that starts at the origin and passes through $(1, 1)$. A line through the origin can only cut it if it also passes through its endpoints, which is forbidden. Valid orientations exist, but none of them cuts the cable; the answer is 0.

Notes

Problem K. Signals That Don't Wait

Time limit: 2000 ms

Memory limit: 256 MiB

An isolated observatory has an antenna powered by a battery. During the next n days the observatory will generate one message per day. Each message can only be transmitted on the day it is generated; if it is not transmitted, it is lost forever.

The battery is initially empty and its available capacity may change from one day to another.

For each day i three integers are known:

- r_i : the amount of energy the generator delivers that day.
- c_i : the maximum capacity of the battery that day.
- a_i : the energy needed to transmit that day's message.

If the battery contains E units of energy at the end of the previous day, day i happens in this order:

1. The capacity is adjusted and the recharge takes place: the available energy becomes $E = \min(c_i, E + r_i)$. All energy exceeding the capacity is lost for good.
2. The day's message can be transmitted if $E \geq a_i$. In that case exactly a_i units of energy are consumed. It is also possible to choose not to transmit it, even if there is enough energy.
3. The remaining energy is kept for the next day.

It is not possible to transmit before the recharge, to save a message for another day, or to transmit it partially. There are no other energy consumptions.

The data for all days is known from the beginning.

Determine the maximum number of messages that can be transmitted during the n days.

Input

The first line contains an integer t ($1 \leq t \leq 10^4$), the number of test cases.

For each test case:

The first line contains an integer n ($1 \leq n \leq 2 \times 10^5$).

Each of the following n lines contains three integers r_i, c_i, a_i ($0 \leq r_i \leq 10^9$, $1 \leq c_i \leq 10^9$, $1 \leq a_i \leq 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed 2×10^5 .

The capacity c_i may increase or decrease from one day to another. It is not guaranteed that $a_i \leq c_i$.

Output

For each test case, print an integer: the maximum number of messages that can be transmitted.

Examples

Sample input 1	Sample output 1
<pre> 4 3 10 10 10 3 10 9 0 10 2 3 6 6 6 0 6 3 0 6 3 5 8 8 5 0 2 2 4 6 3 1 6 2 0 6 1 2 0 5 1 2 1 2 </pre>	<pre> 2 2 4 0 </pre>

Explanation of sample 1

In the first case, the messages of days 1 and 3 can be transmitted. After the first transmission the battery is empty; on the second day it receives 3 units, which are kept for the third day. The three messages cannot all be transmitted because, after sending the first one, there is not enough energy left for the second.

In the second case it is better not to transmit the first message: this way the initial 6 units allow transmitting the messages of days 2 and 3.

In the third case the messages of days 1, 2, 3 and 4 can be transmitted. The energy at the end of each of those days is, respectively, 3, 0, 1 and 0. In particular, at the start of the second day the capacity reduction makes the available energy drop from 3 to 2. The fifth message cannot be added to this plan: transmitting all five messages is impossible.

In the fourth case, there is no energy on the first day. On the second day, although the generator delivers 2 units, the battery can only store 1, so the message cannot be transmitted either.

Notes