

2026 EGOI Team Formation Test

Task Overview

ID	Name	Time Limit	Memory Limit	Subtasks
T26G1	Digit Puzzle II	1.000 s	256 MB	8 + 10 + 22 + 13 + 17 + 24 + 6
T26G2	Closing Ceremony	1.000 s	256 MB	7 + 9 + 21 + 23 + 15 + 11 + 14
T26G3	Annoying Advertisements II	2.000 s	256 MB	19 + 12 + 20 + 26 + 23
T26G4	Mountains	2.500 s	256 MB	11 + 22 + 16 + 39 + 12

Notice:

Unless otherwise specified, inputs and outputs shall follow the format below:

- One space between a number and another number or character in the same line.
- No space between characters in the same line.
- Each string shall be placed in its own separate line.
- Outputs will be automatically fixed as follows: Trailing spaces in each line will be removed and an end-of-line character will be added to the end of the output if not present. All other format errors will not be fixed.

C++ programmers should be aware that using C++ streams (`cin` / `cout`) may lead to I/O bottlenecks and substantially lower performance.

For some problems 64-bit integers may be required. In C++ it is `long long` and its token for `scanf`/`printf` is `%lld`.

All tasks are divided into subtasks. You need to pass all test cases in a subtask to get points.

T26G1 - DIGIT PUZZLE II

Time Limit: 1.000 s / Memory Limit: 256 MB

Alice remembers that [Bob enjoys solving digit puzzles](#), so she decided to come up with her own variant to solve as well.

Bob's version of the puzzle consists of $N + 1$ blank cells separated by N arithmetic operators, which are either $+$, $-$, or $*$. Bob must then fill the blank cells with numbers to satisfy the equation. However, Alice thinks that Bob's version takes away all the fun; she believes that choosing the arithmetic operators is much more interesting and satisfying than writing down numbers. Consequently, Alice inverted Bob's logic and revamped the rules. Her own variant of the puzzle works as follows:

Each puzzle contains $N + 1$ prefilled numbers, where every two adjacent numbers are separated by a blank space, resulting in N blank spaces in total. Note that while the $N + 1$ prefilled numbers can be positive, negative, or zero, **it is guaranteed that they are all even integers**. We denote the i -th prefilled number from left to right as A_i , where $1 \leq i \leq N + 1$.

The following example shows a puzzle with $N = 5$, $A_1 = 2$, $A_2 = -4$, $A_3 = 0$, $A_4 = 6$, $A_5 = -2$, and $A_6 = 4$:

2		-4		0		6		-2		4
---	--	----	--	---	--	---	--	----	--	---

Afterwards, Alice writes down exactly one arithmetic operator $+$, $-$, or $*$ into each of the first $N - 1$ blank spaces. She also writes down an equal sign into the last blank space. Alice has to ensure that the prefilled numbers, when combined with the arithmetic operators that she writes down, form a mathematically correct equation. In the example above, Alice can achieve this by writing the operators $*$, $+$, $-$, $*$, and $=$ into the spaces in this order to obtain a valid equality, since $2 \times (-4) + 0 - 6 \times (-2) = 4$.

2	*	-4	+	0	-	6	*	-2	=	4
---	---	----	---	---	---	---	---	----	---	---

Unfortunately, just as Alice was ready to solve the puzzle, she discovered that the last number of the puzzle A_{N+1} is missing! Her puzzle now looks like this:

2		-4		0		6		-2		??
---	--	----	--	---	--	---	--	----	--	----

Despite being annoyed, Alice thinks it would be a pity if she could not solve the puzzle at all just because one number A_{N+1} has gone missing. Therefore, she decides to adapt by changing her rules. Not only does Alice choose the arithmetic operations to be filled, she now also chooses the value of A_{N+1} herself. Alice's target now is to fill each of the first $N - 1$ blank spaces with the arithmetic operators $+$, $-$, or $*$, and the last blank space with $=$, so that the resulting value of A_{N+1} is maximised. Of course, the equation must still be mathematically correct.

Your task is to help Alice find the largest possible integer A_{N+1} that can be formed by filling each of the first $N - 1$ blank spaces with $+$, $-$, or $*$, and the last blank space with $=$, so that the resulting equality holds.

All test cases are generated so that no matter what operators Alice chooses to write down into the first $N - 1$ blank spaces, the value of the evaluated expression (using the standard order of operations: multiplications before additions and subtractions) is always between -10^9 and 10^9 inclusive.



INPUT

The first line of the input contains a single positive integer N .

The second line of the input contains N **even** integers, $A_1, A_2, \dots, A_N$, denoting the first N prefilled numbers of the puzzle in order from left to right.

OUTPUT

Output one single integer, the largest possible value of the missing integer A_{N+1} .

SAMPLE TESTS

	Input	Output			
1	<table><tr><td>5</td></tr><tr><td>2 -4 0 6 -2</td></tr></table>	5	2 -4 0 6 -2	<table><tr><td>18</td></tr></table>	18
5					
2 -4 0 6 -2					
18					

Note that $2 - (-4) - 0 - 6 \times (-2) = 18$.

It can be proven that it is impossible to form a valid equality whenever A_{N+1} is an integer larger than 18.

2	<table><tr><td>5</td></tr><tr><td>-4 10 0 6 -8</td></tr></table>	5	-4 10 0 6 -8	<table><tr><td>54</td></tr></table>	54
5					
-4 10 0 6 -8					
54					

SUBTASKS

For all cases:

$$2 \leq N \leq 2 \times 10^5$$

$$-100 \leq A_i \leq 100 \text{ for every } 1 \leq i \leq N$$

$$A_i \text{ is even for every } 1 \leq i \leq N$$

All test cases are generated so that no matter what operators Alice chooses, the value of the evaluated expression is between -10^9 and 10^9 inclusive

	Points	Constraints
1	8	$N = 2$ $A_1 = 0$
2	10	$A_i \geq 2$ for every $1 \leq i \leq N$
3	22	$A_i \geq 0$ for every $1 \leq i \leq N$
4	13	$A_1 = 0$
5	17	N is odd $A_i = -2$ for every $1 \leq i \leq N$
6	24	N is odd $A_i \leq -2$ for every $1 \leq i \leq N$
7	6	No additional constraints

T26G2 - CLOSING CEREMONY

Time Limit: 1.000 s / Memory Limit: 256 MB

You are now at the closing ceremony of EGOI 2026 in Cesenatico, Italy, and the medallists are queueing to go up to the stage and receive their prizes. You had a solution to the third problem of the second contest day, but you didn't manage to submit it in time. You feel terrible by the fact that you could have earned 50 more points, not to mention you just learnt that you were a mere 2 points away from the bronze medal cutoff... That sucks. That really sucks.

Feeling helpless, you sit in your seat and contemplate while watching the medallists with huge smiles and laughter on their faces. "What if I could make them leave the queue?" you murmur to yourself, hoping you would be less irritated by the celebratory mood.

You counted that there are N medallists in the queue. For each $1 \leq i \leq N$, the i -th person from the front received exactly one type of medal: gold, silver, or bronze.

A subsegment consisting of the l -th to the r -th medallists among the queue, where $1 \leq l \leq r \leq N$, is called "vanishing", if it is possible for everyone in that subsegment to leave the queue by repeatedly performing the following operation (until no one from the subsegment remains):

- Two medallists within the subsegment who are currently standing immediately next to each other and obtained different types of medals (gold, silver, or bronze) get into a heated argument about which medal design is more beautiful, and they both leave the queue. The remaining medallists in the queue then move to fill up the gap in the queue.

For example, the subsegment GSSBBB is vanishing because it is possible to perform the following operations in order:

- Remove the third and fourth medallists (who obtained silver and bronze respectively), and then the subsegment becomes GSBB.
- Remove the second and third medallists (who obtained silver and bronze respectively), and then the subsegment becomes GB.
- Remove the first and second medallists (who obtained gold and bronze respectively), and then the subsegment becomes empty.

On the other hand, it can be proven that the subsegments SSBBB and BB are not vanishing.

Bored by the long closing ceremony, you decided to count how many different vanishing subsegments there are in the queue. Here, a subsegment consisting of the l_1 -th to the r_1 -th medallists is regarded as different to the subsegment consisting of the l_2 -th to the r_2 -th medallists if and only if at least one of $l_1 \neq l_2$ or $r_1 \neq r_2$ hold true. In particular, even if two subsegments are composed with medallists with the same medal order, as long as their starting and ending indices are not identical, they are regarded as different subsegments. Also, the medallists who left the queue as a result of an operation while considering the current subsegment all rejoin the queue at their initial positions, before you start to determine whether another subsegment is vanishing or not.

...and then you wake up. You look at your phone: Saturday, 16 May, 07:00 AM, the second contest day of EGOI 2026. It was all just a nightmare. You still have a bit more than two hours before you arrive at the contest venue and let fate decide which medal colour you are going to receive. Hoping that the nightmare won't actually come true, you decide to code out the solution to this problem to make sure you are ready for the contest.

INPUT

The first line of the input contains a single positive integer N , the number of medallists in the queue.

The second line of the input contains a string of length N consisting of the characters **G**, **S**, and **B** only. The i -th character is **G** if the i -th person from the front received a gold medal, **S** for a silver medal, and **B** for a bronze medal.

OUTPUT

Output a single non-negative integer, the number of vanishing subsegments in the queue.

SAMPLE TESTS

	Input	Output
1	4 GSBS	4
	The vanishing subsegments are GS , SB , BS , and GSBS . It can be proven that all other subsegments are not vanishing.	
2	6 GSSBBB	5
	The vanishing subsegments are GS , SB , GSSB , SSBB , and GSSBBB .	
3	7 BSSBBBS	6
	The vanishing subsegments are BS which occurs twice (once with $l = 1, r = 2$, and once with $l = 6, r = 7$), BSSB , SB , SSBB , and SSBBBS .	

SUBTASKS

For all cases: $1 \leq N \leq 2 \times 10^5$

The input string contains the characters **G**, **S**, and **B** only

	Points	Constraints
1	7	N is even The input string is of the form BSBSBS...BS
2	9	$N = 4$
3	21	$N \leq 500$ The input string contains the characters S and B only
4	23	The input string contains the characters S and B only
5	15	$N \leq 5000$
6	11	The input string does not contain two consecutive S characters The input string does not contain two consecutive B characters
7	14	No additional constraints

T26G3 - ANNOYING ADVERTISEMENTS II

Time Limit: 2.000 s / Memory Limit: 256 MB

Oh no! It's that runner mobile game advertisement again! Being annoyed, you want to write a program to beat it so you don't have to play it yourself.

In this runner game, your goal is to gather the largest number of runners possible by changing colors and passing through gates. There are K colors in total, numbered from 1 to K . You start off with a single uncolored runner. Then the game will proceed for N rounds, alternating between Coloring Rounds and Reward Rounds. In the i -th round ($1 \leq i \leq N$):

- If i is odd, this is a Coloring Round. You will be facing C_i color gates where the j -th gate is of color $A_{i,j}$. If the chosen gate is uncolored (represented by $A_{i,j} = 0$), the color of your runners will remain unchanged. Otherwise, the color of your runners will be changed to $A_{i,j}$.
- If i is even, this is a Reward Round. You will be facing C_i reward gates where the j -th gate has a color $B_{i,j}$ and an integer $X_{i,j}$. If your runners are of color $B_{i,j}$ when passing through the j -th gate (that is, the current color of your runners match that of the gate), you will gain another $X_{i,j}$ runners that are also of color $B_{i,j}$. Otherwise, you gain nothing. You may refer to Figure 1, 2, 3 for more details.

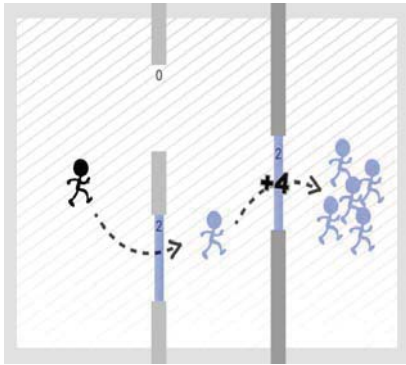


Figure 1: Passing through a reward gate with the same color, get $X_{i,j}$ more runners of the same color

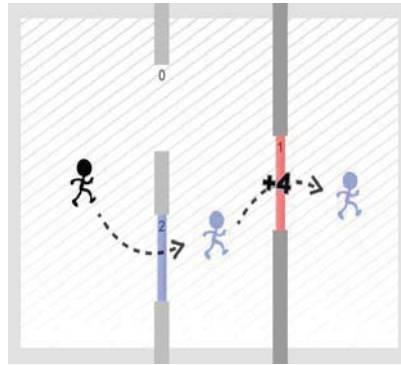


Figure 2: Passing through a reward gate with a different color, don't get any runners

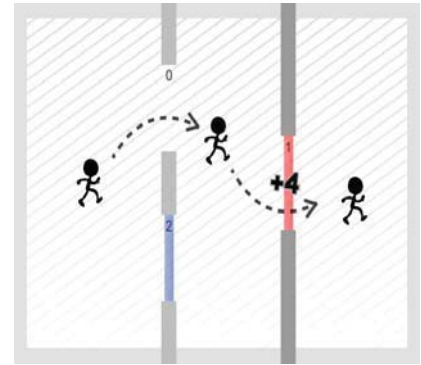


Figure 3: Uncolored runners passing through a reward gate, also don't get any runners

Note that in every round, you must choose **exactly one** gate to pass through.

Given the information of the gates in the N rounds, find the maximum number of runners you can end up with after all N rounds.

INPUT

The first line of the input contains two integers N and K , the total number of rounds and the number of colors.

Then, N lines follow. The i -th line starts with an integer C_i .

If i is odd, then C_i integers follow, $A_{i,1}, A_{i,2}, \dots, A_{i,C_i}$, the color of the gates.

If i is even, then $2 \times C_i$ integers follow, $B_{i,1}, X_{i,1}, B_{i,2}, X_{i,2}, \dots, B_{i,C_i}, X_{i,C_i}$, the color and integer of the gates.

OUTPUT

Output a single integer, the maximum number of runners you can end up with after all N rounds.

SAMPLE TESTS

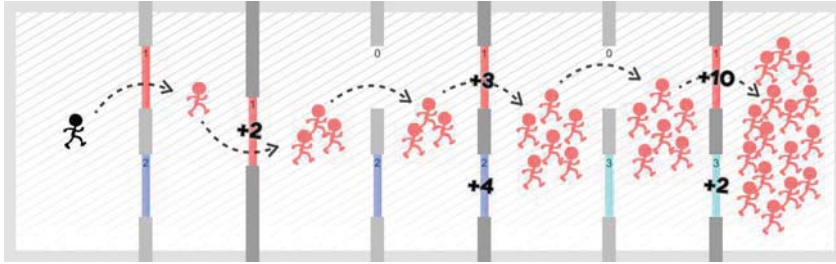
Input

Output

1

6 3
2 1 2
1 1 2
2 0 2
2 1 3 2 4
2 0 3
2 1 10 3 2

16



The following is the optimal way:

- Round 1 (Coloring Round): Choose the gate with color 1, changed to color 1.
- Round 2 (Reward Round): Choose the gate with color 1, gained 2 runners.
- Round 3 (Coloring Round): Choose the uncolored gate, color remained unchanged as 1.
- Round 4 (Reward Round): Choose the gate with color 1, gained 3 runners.
- Round 5 (Coloring Round): Choose the uncolored gate, color remained unchanged as 1.
- Round 6 (Reward Round): Choose the gate with color 1, gained 10 runners.

Total number of runners: $1 + 2 + 3 + 10 = 16$.

2

4 4
4 1 2 3 4
2 1 3 3 6
2 0 2
3 2 5 3 4 4 10

12

SUBTASKS

For all cases:

$$2 \leq N \leq 10^5$$

$$1 \leq K \leq 10^6$$

$$1 \leq C_i \text{ for every } 1 \leq i \leq N$$

$$\text{Sum of } C_i \leq 5 \times 10^5$$

For every $1 \leq i \leq N$ where i is odd:

- $0 \leq A_{i,j} \leq K$ for every $1 \leq j \leq C_i$
- $A_{i,j_1} \neq A_{i,j_2}$ for every $1 \leq j_1 < j_2 \leq C_i$

For every $1 \leq i \leq N$ where i is even:

- $1 \leq B_{i,j} \leq K$ for every $1 \leq j \leq C_i$
- $1 \leq X_{i,j} \leq 10^4$ for every $1 \leq j \leq C_i$
- $B_{i,j_1} \neq B_{i,j_2}$ for every $1 \leq j_1 < j_2 \leq C_i$

Points

Constraints

1

19

$A_{i,j} \neq 0$
Sum of $C_i \leq 1000$

2

12

$A_{i,j} \neq 0$

3

20

$A_{i,j} \neq 0$ for all $i \equiv 1 \pmod{4}$

4

26

$K \leq 10$

5

23

No additional constraints

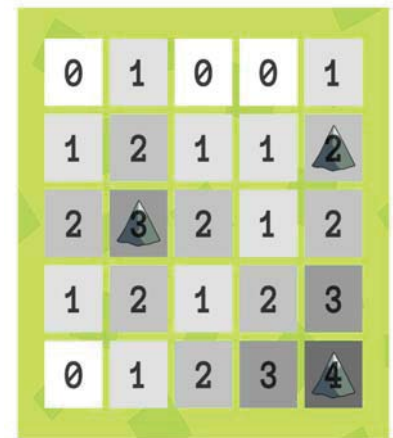
T26G4 - MOUNTAINS

Time Limit: 2.500 s / Memory Limit: 256 MB

There once was a foolish old man who lived among the mountains. Every time he wanted to travel to the city, he had to hike over steep peaks and deep valleys. One day, having had enough, he set himself an impossible goal: to level the mountains.

The old man lives on a $N \times N$ grid, where the cell (u, v) is located at row u and column v ($1 \leq u, v \leq N$). On the grid, there are M mountains. The i -th mountain has its peak at cell (X_i, Y_i) with height H_i . No two mountains have peaks on the same cell.

The height of any cell is relative to the distance of the nearest mountain, where the height of cells decrease the further away it is from the peak of a mountain. Formally, the height of cell (u, v) is equal to the maximum of $\max(0, H_i - |X_i - u| - |Y_i - v|)$ among all mountains i . Note that this means the height of every cell is a non-negative integer. By this definition, it is also possible that the height of cell (X_i, Y_i) is greater than H_i if some taller mountain is nearby.



(A 5×5 grid with $M = 3$ mountains)

The old man wants to level the mountains such that every cell has the same height, it does not matter which exact height the cells are levelled to as long as they are all equal.

Every day, the old man may level one cell by increasing or decreasing its height by 1. Note that levelling a cell (u, v) does not affect the height of any other cells, even if (u, v) was originally a mountain peak.

You are astonished by the old man's dedication, so you want to help the old man calculate the optimal height the cells should be levelled to, in order to minimise the number of days required to level all cells to the same height.

INPUT

The first line contains two integers N and M , the size of the grid and the number of mountains.

The next M lines contain three integers. The i -th line contains X_i , Y_i and H_i , the cell and height of the i -th mountain peak.

OUTPUT

Output a single integer, the optimal height the cells should be levelled to for the minimum number of days. If there are multiple answers, you may output any of them.

SAMPLE TESTS

Input

Output

1

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

This sample corresponds to the figure in the statement.

- Levelling all the cells to height 0 would take 38 days
- Levelling all the cells to height 1 would take 21 days
- Levelling all the cells to height 2 would take 22 days
- Levelling all the cells to height 3 would take 39 days

It can be proven that if all the cells are levelled to a height greater than 3, then it would not result in a minimum number of days.

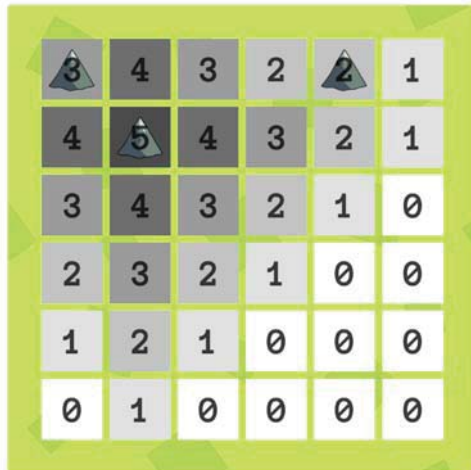
If all the cells were levelled to height 1, it would take the old man 21 days, which is the minimum.

2

6 3	1
1 1 2	
2 2 5	
1 5 2	

The following figure shows the initial heights of the grid.

Note that the initial height of (1, 1) is actually $5 - |2 - 1| - |2 - 1| = 3$ due to Mountain 2.



3

6 2	2
2 2 5	
1 5 2	

The heights of every cell in Sample 3 is the same as Sample 2, in which both heights 1 and 2 result in a minimum number of days.

SUBTASKS

For all cases:

$$1 \leq N \leq 10^9$$

$$1 \leq M \leq 5 \times 10^4$$

$$1 \leq X_i, Y_i \leq N \text{ for every } 1 \leq i \leq M$$

$$(X_i, Y_i) \neq (X_j, Y_j) \text{ for every } 1 \leq i, j \leq M, i \neq j$$

$$1 \leq H_i \leq 10^6 \text{ for every } 1 \leq i \leq M$$

	Points	Constraints
1	11	$N, M \leq 100$ $H_i \leq 100$ for every $1 \leq i \leq M$
2	22	$N, M \leq 100$
3	16	$N, M \leq 2000$
4	39	For all $1 \leq i \leq N$, the cells on the boundary of the grid which are $(1, i)$, (N, i) , $(i, 1)$, (i, N) have a height of 0
5	12	No additional constraints