Codeforces Beta Round #87 (Div. 1 Only)

A. Party

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A company has n employees numbered from 1 to n. Each employee either has no immediate manager or exactly one immediate manager, who is another employee with a different number. An employee A is said to be the *superior* of another employee B if at least one of the following is true:

- Employee A is the immediate manager of employee B
- Employee B has an immediate manager employee C such that employee A is the superior of employee C.

The company will not have a managerial cycle. That is, there will not exist an employee who is the superior of his/her own immediate manager.

Today the company is going to arrange a party. This involves dividing all n employees into several groups: every employee must belong to exactly one group. Furthermore, within any single group, there must not be two employees A and B such that A is the superior of B.

What is the minimum number of groups that must be formed?

Input
The first line contains integer n ($1 \le n \le 2000$) — the number of employees.

The next n lines contain the integers p_i ($1 \le p_i \le n$ or $p_i = -1$). Every p_i denotes the immediate manager for the i-th employee. If p_i is -1, that means that the i-th employee does not have an immediate manager.

It is guaranteed, that no employee will be the immediate manager of him/herself ($p_i \neq i$). Also, there will be no managerial cycles.

Output
Print a single integer denoting the minimum number of groups that will be formed in the party.

Examples

input
5 -1 1 2 1 -1
output
3

Note
For the first example, three groups are sufficient, for example:

- Employee 1
- Employees 2 and 4
- Employees 3 and 5

B. Lawnmower

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a garden consisting entirely of grass and weeds. Your garden is described by an $n \times m$ grid, with rows numbered 1 to n from top to bottom, and columns 1 to m from left to right. Each cell is identified by a pair (r, c) which means that the cell is located at row r and column c. Each cell may contain either grass or weeds. For example, a 4×5 garden may look as follows (empty cells denote grass):

You have a land-mower with you to mow all the weeds. Initially, you are standing with your lawnmower at the top-left corner of the garden. That is, at cell $(1, 1)$. At any moment of time you are facing a certain direction — either left or right. And initially, you face right.

In one move you can do either one of these:

1) Move one cell in the direction that you are facing.

- if you are facing right: move from cell (r, c) to cell $(r, c + 1)$

- if you are facing left: move from cell (r, c) to cell $(r, c - 1)$

2) Move one cell down (that is, from cell (r, c) to cell $(r + 1, c)$), and change your direction to the opposite one.

- if you were facing right previously, you will face left

- if you were facing left previously, you will face right

You are not allowed to leave the garden. Weeds will be mowed if you and your lawnmower are standing at the cell containing the weeds (your direction doesn't matter). This action isn't counted as a move.

What is the minimum number of moves required to mow all the weeds?

Input

The first line contains two integers n and m ($1 \le n, m \le 150$) — the number of rows and columns respectively. Then follow n lines containing m characters each — the content of the grid. "G" means that this cell contains grass. "W" means that this cell contains weeds.

It is guaranteed that the top-left corner of the grid will contain grass.

Output

Print a single number — the minimum number of moves required to mow all the weeds.

Examples

input
4 5 GWGGW GGWGG GWGGG WGGGG
output
11

input
3 3 GWW WWW WWG
output
7

input
1 1

G
output
0

Note

For the first example, this is the picture of the initial state of the grid:

A possible solution is by mowing the weeds as illustrated below:

C. Plumber

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Little John aspires to become a plumber! Today he has drawn a grid consisting of n rows and m columns, consisting of $n \times m$ square cells.

In each cell he will draw a pipe segment. He can only draw four types of segments numbered from 1 to 4, illustrated as follows:

Each pipe segment has two ends, illustrated by the arrows in the picture above. For example, segment 1 has ends at top and left side of it.

Little John considers the piping system to be leaking if there is at least one pipe segment inside the grid whose end is not connected to another pipe's end or to the border of the grid. The image below shows an example of leaking and non-leaking systems of size 1×2.

Now, you will be given the grid that has been partially filled by Little John. Each cell will either contain one of the four segments above, or be empty. Find the number of possible different non-leaking final systems after Little John finishes filling **all** of the empty cells with pipe segments. Print this number modulo 1000003 ($10^6 + 3$).

Note that rotations or flipping of the grid are not allowed and so two configurations that are identical only when one of them has been rotated or flipped either horizontally or vertically are considered two different configurations.

Input

The first line will contain two single-space separated integers n and m ($1 \le n, m, n \cdot m \le 5 \cdot 10^5$) — the number of rows and columns respectively. Then n lines follow, each contains exactly m characters — the description of the grid. Each character describes a cell and is either one of these:

- "1" - "4" — a pipe segment of one of four types as described above
- "." — an empty cell

Output

Print a single integer denoting the number of possible final non-leaking pipe systems modulo 1000003 ($10^6 + 3$). If there are no such configurations, print 0.

Examples

input
2 2 13 ..
output
2

input
3 1 1 4 .
output
0

input
2 2 3. .1
output
1

Note

For the first example, the initial configuration of the grid is as follows.

The only two possible final non-leaking pipe configurations are as follows:

For the second example, the initial grid is already leaking, so there will be no final grid that is non-leaking.

For the final example, there's only one possible non-leaking final grid as follows.

D. Unambiguous Arithmetic Expression

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's define an *unambiguous arithmetic expression* (UAE) as follows.

- All non-negative integers are UAE's. Integers may have leading zeroes (for example, 0000 and 0010 are considered valid integers).
- If X and Y are two UAE's, then "$(X) + (Y)$", "$(X) - (Y)$", "$(X) * (Y)$", and "$(X) / (Y)$" (all without the double quotes) are UAE's.
- If X is an UAE, then "$-(X)$" and "$+(X)$" (both without the double quotes) are UAE's.
  You are given a string consisting only of digits ("0" - "9") and characters "-", "+", "*", and "/". Your task is to compute the number of different possible unambiguous arithmetic expressions such that if all brackets (characters "(" and ")") of that unambiguous arithmetic expression are removed, it becomes the input string. Since the answer may be very large, print it modulo 1000003 ($10^6 + 3$).

Input

The first line is a *non-empty* string consisting of digits ('0'-'9') and characters '-', '+', '*', and/or '/'. Its length will not exceed 2000. The line doesn't contain any spaces.

Output

Print a single integer representing the number of different unambiguous arithmetic expressions modulo 1000003 ($10^6 + 3$) such that if all its brackets are removed, it becomes equal to the input string (character-by-character).

Examples

input
1+2*3
output
2

input
03+-30+40
output
3

input
5//4
output
0

input
5/0
output
1

input
1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1
output
100728

Note

For the first example, the two possible unambiguous arithmetic expressions are:

$((1) + (2)) * (3)$
$(1) + ((2) * (3))$

For the second example, the three possible unambiguous arithmetic expressions are:

$(03) + ((- (30)) + (40))$
$(03) + (- ((30) + (40)))$
$((03) + (- (30))) + (40)$

E. Linear Kingdom Races

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are a car race organizer and would like to arrange some races in Linear Kingdom.

Linear Kingdom has n consecutive roads spanning from left to right. The roads are numbered from 1 to n from left to right, thus the roads follow in the order of their numbers' increasing. There will be several races that may be held on these roads. Each race will use a **consecutive** subset of these roads. Also, each race will pay some amount of money to you if this race is held. No races overlap in time, so some roads can be used in several races.

Unfortunately, some of the roads are in a bad condition and they need repair. Each road has repair costs associated with it, you are required to pay this cost to repair the road. A race can only take place if all the roads used in the race are renovated. Your task is to repair such roads (possibly all or none) that will maximize your profit. Your profit is defined as the total money you get from the races that are held minus the total money you spent to repair the roads. Note that you may decide not to repair any road and gain zero profit.

Print the maximum profit you can gain.

Input

The first line contains two single-space separated integers, n and m ($1 \le n, m \le 2 \cdot 10^5$), denoting the number of roads and the number of races, respectively.

Then n lines follow, each line will contain a single non-negative integer not exceeding 10^9 denoting the cost to repair a road. The costs are given in order from road 1 to road n.

Finally, m lines follow. Each line is single-space-separated triplets of integers. Each triplet will be given as lb, ub, and p ($1 \le lb \le ub \le n, 1 \le p \le 10^9$), which means that the race these three integers describe will use all the roads from lb to ub, inclusive, and if it's held you get p.

Output

Print a single integer denoting the maximum possible profit you can gain.

Please, do not use the `%lld` specificator to read or write 64-bit integers in C++. It is recommended to use `cin`, `cout` stream (also you may use `%I64d` specificator).

Examples

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

input
2 1 0 3 1 2 5
output
2

input
3 1 10 10 10 1 3 10
output
0

Note

In the first sample the optimal solution is to repair roads 1, 2, 3, and 7. Three races will take place which nets you 15. The road repair costs 11, hence your profit is 4.
