A. Collecting Beats is Fun

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Cucumber boy is fan of Kyubeat, a famous music game.

Kyubeat has 16 panels for playing arranged in 4×4 table. When a panel lights up, he has to press that panel.

Each panel has a **timing** to press (the preffered time when a player should press it), and Cucumber boy is able to press at most k panels in a time with his one hand. Cucumber boy is trying to press all panels in perfect timing, that is he wants to press each panel exactly in its preffered time. If he cannot press the panels with his **two hands** in perfect timing, his challenge to press all the panels in perfect timing will fail.

You are given one scene of Kyubeat's panel from the music Cucumber boy is trying. Tell him is he able to press all the panels in perfect timing.

Input

The first line contains a single integer k $(1 \le k \le 5)$ — the number of panels Cucumber boy can press with his one hand.

Next 4 lines contain 4 characters each (digits from 1 to 9, or period) — table of panels. If a digit i was written on the panel, it means the boy has to press that panel in time i. If period was written on the panel, he doesn't have to press that panel.

Output

Output "YES" (without quotes), if he is able to press all the panels in perfect timing. If not, output "NO" (without quotes).

Sample test(s)

input
```
1
.135
1247
3468
5789
```
output
```
YES
```

input
```
5
..1.
1111
..1.
..1.
```
output
```
YES
```

input
```
1
....
12.1
.2..
.2..
```
output
```
NO
```

Note

In the third sample boy cannot press all panels in perfect timing. He can press all the panels in timing in time 1, but he cannot press the panels in time 2 in timing with his two hands.

B. Making Sequences is Fun

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

We'll define $S(n)$ for positive integer n as follows: the number of the n's digits in the decimal base. For example, $S(893) = 3$, $S(114514) = 6$.

You want to make a consecutive integer sequence starting from number m (m, $m + 1$, ...). But you need to pay $S(n) \cdot k$ to add the number n to the sequence.

You can spend a cost up to w, and you want to make the sequence as long as possible. Write a program that tells sequence's maximum length.

Input

The first line contains three integers w ($1 \le w \le 10^{16}$), m ($1 \le m \le 10^{16}$), k ($1 \le k \le 10^9$).

Please, do not write the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Output

The first line should contain a single integer — the answer to the problem.

Sample test(s)

input
```
9 1 1
```
output
```
9
```

input
```
77 7 7
```
output
```
7
```

input
```
114 5 14
```
output
```
6
```

input
```
1 1 2
```
output
```
0
```

C. Counting Kangaroos is Fun

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

There are n kangaroos with pockets. Each kangaroo has a size (integer number). A kangaroo can go into another kangaroo's pocket if and only if the size of kangaroo who hold the kangaroo is at least twice as large as the size of kangaroo who is held.

Each kangaroo can hold at most one kangaroo, and the kangaroo who is held by another kangaroo cannot hold any kangaroos.

The kangaroo who is held by another kangaroo cannot be visible from outside. Please, find a plan of holding kangaroos with the minimal number of kangaroos who is visible.

Input

The first line contains a single integer — n $(1 \le n \le 5 \cdot 10^5)$. Each of the next n lines contains an integer s_i — the size of the i-th kangaroo $(1 \le s_i \le 10^5)$.

Output

Output a single integer — the optimal number of visible kangaroos.

Sample test(s)

input

```
8
2
5
7
6
9
8
4
2
```

output

```
5
```

input

```
8
9
1
6
2
6
5
8
3
```

output

```
5
```

D. Counting Rectangles is Fun

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is an $n \times m$ rectangular grid, each cell of the grid contains a single integer: zero or one. Let's call the cell on the i-th row and the j-th column as (i, j).

Let's define a "rectangle" as four integers a, b, c, d $(1 \le a \le c \le n;\ 1 \le b \le d \le m)$. Rectangle denotes a set of cells of the grid $\{(x, y) :\ a \le x \le c, b \le y \le d\}$. Let's define a "good rectangle" as a rectangle that includes only the cells with zeros.

You should answer the following q queries: calculate the number of good rectangles all of which cells are in the given rectangle.

Input

There are three integers in the first line: n, m and q $(1 \le n, m \le 40,\ 1 \le q \le 3 \cdot 10^5)$. Each of the next n lines contains m characters — the grid. Consider grid rows are numbered from top to bottom, and grid columns are numbered from left to right. Both columns and rows are numbered starting from 1.

Each of the next q lines contains a query — four integers that describe the current rectangle, a, b, c, d $(1 \le a \le c \le n;\ 1 \le b \le d \le m)$.

Output

For each query output an answer — a single integer in a separate line.

Sample test(s)

input

```
5 5 5
00101
00000
00001
01000
00001
1 2 2 4
4 5 4 5
1 2 5 2
2 2 4 5
4 2 5 3
```

output

```
10
1
7
34
5
```

input

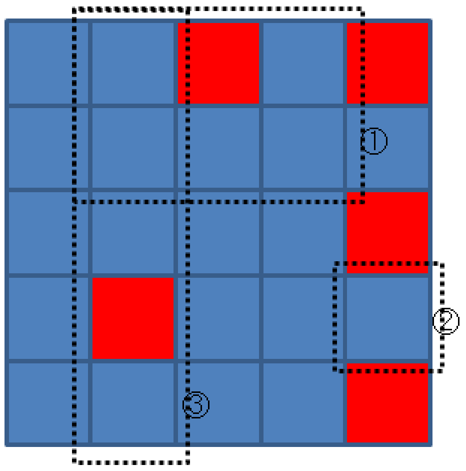
```
4 7 5
0000100
0000010
0011000
0000000
1 7 2 7
3 1 3 1
2 3 4 5
1 2 2 7
2 2 4 7
```

output

```
3
1
16
27
52
```

Note

For the first example, there is a 5×5 rectangular grid, and the first, the second, and the third queries are represented in the following image.



- For the first query, there are 10 good rectangles, five 1×1, two 2×1, two 1×2, and one 1×3.
- For the second query, there is only one 1×1 good rectangle.
- For the third query, there are 7 good rectangles, four 1×1, two 2×1, and one 3×1.

E. Watching Fireworks is Fun

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A festival will be held in a town's main street. There are n sections in the main street. The sections are numbered 1 through n from left to right. The distance between each adjacent sections is 1.

In the festival m fireworks will be launched. The i-th $(1 \le i \le m)$ launching is on time t_i at section a_i. If you are at section x $(1 \le x \le n)$ at the time of i-th launching, you'll gain happiness value $b_i - |a_i - x|$ (note that the happiness value might be a negative value).

You can move up to d length units in a unit time interval, but it's prohibited to go out of the main street. Also you can be in an arbitrary section at initial time moment (time equals to 1), and want to maximize the sum of happiness that can be gained from watching fireworks. Find the maximum total happiness.

Note that two or more fireworks can be launched at the same time.

Input

The first line contains three integers n, m, d $(1 \le n \le 150000;\ 1 \le m \le 300;\ 1 \le d \le n)$.

Each of the next m lines contains integers a_i, b_i, t_i $(1 \le a_i \le n;\ 1 \le b_i \le 10^9;\ 1 \le t_i \le 10^9)$. The i-th line contains description of the i-th launching.

It is guaranteed that the condition $t_i \le t_{i+1}$ $(1 \le i < m)$ will be satisfied.

Output

Print a single integer — the maximum sum of happiness that you can gain from watching all the fireworks.

Please, do not write the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Sample test(s)

input
```
50 3 1
49 1 1
26 1 4
6 1 10
```
output
```
-31
```

input
```
10 2 1
1 1000 4
9 1000 4
```
output
```
1992
```