Problem A. CosmoTile

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

In the year 3030, hamsters decided to improve their spaceship. Specifically, they wanted to lay tiles on the floor.

The floor is square with a side length of a meters. Rectangular tiles will be laid on it, and:

- the sides of the tiles must be parallel to the sides of the floor;
- all tiles must be oriented the same way;
- the tiles will be laid tightly against each other;
- the first tile will be laid tightly against the "top left" corner of the floor.

For a better understanding of the tile arrangement, see the illustration in the example.

The hamsters have at their disposal:

- a 3D blueprint of the rectangular tile with dimensions a meters by b meters (the thickness is so small that it can be neglected);
- a 3D printer that can print any number of tiles;
- an ultra-thin and ultra-precise laser that can cut tiles without leaving any waste;
- a reducing laser that can reduce any tile by any integer x times, but the value of x must be at least 1 and at most k;
- boundless energy to lay the tiles as described above.

The hamsters' plan is as follows:

- Choose a number x;
- Print the minimum necessary number of tiles;
- Cut some tiles so that using the resulting pieces of tile (considering further reduction) they can cover the entire floor. Each tile can be cut no more than once, only along a straight line parallel to the side of length a, and if cut at a distance of r meters from the side of length a, two pieces of tile will result: one with dimensions a by r meters and the other with dimensions a by $b-r$ meters. The resulting pieces cannot be cut further;
- The pieces of tile that will ultimately be laid on the floor should be reduced by x times (that is, both the length and the width of the piece of tile will be reduced exactly by x times);
- Lay the tiles as described above.

The hamsters are economical but maintain symmetry in the pattern. Therefore, if cutting a tile into two pieces results in pieces of the same size, both pieces are used (for better understanding, see the illustration in the example when $x = 3$).

The hamsters have tasked you with choosing x in such a way that the total area of the unused pieces of tile is minimized. What is this minimum total area?

Input

The first line contains the integer a—the length of the side of the floor and the first side of the tile ($1 \le a \le 2 \cdot 10^6$).

The second line contains the integer b—the length of the second side of the tile ($1 \le b \le 2 \cdot 10^6$).

The third line contains the integer k—the maximum value of the parameter x for the reducing laser ($1 \le k \le 2 \cdot 10^{18}$).

Output

Output a single number—the minimum total area of the unused pieces.

Scoring

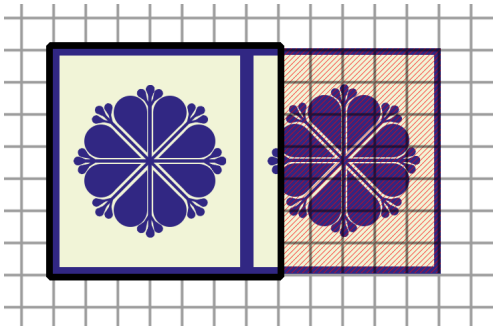
Subtask	Points	Additional Constraints	Required Groups	Comment
0	0	–	–	Tests from the statement.
1	20	$a, b, k \le 200$; b — odd	–	–
2	15	$a, b, k \le 200$	0–1	–
3	15	$a, b, k \le 2\,000$	0–2	–
4	20	$k \le 2 \cdot 10^6$	0–3	–
5	10	$k \le 2 \cdot 10^{12}$	0–4	–
6	20	—	0–5	–

Examples

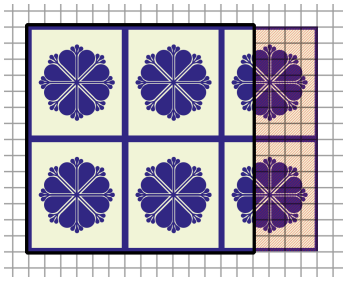
standard input	standard output
7 6 4	21
11 8 4	0

Explanation

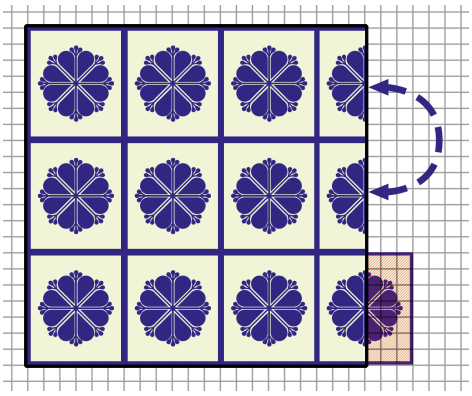
Illustration for the first example for all possible x on the next page.



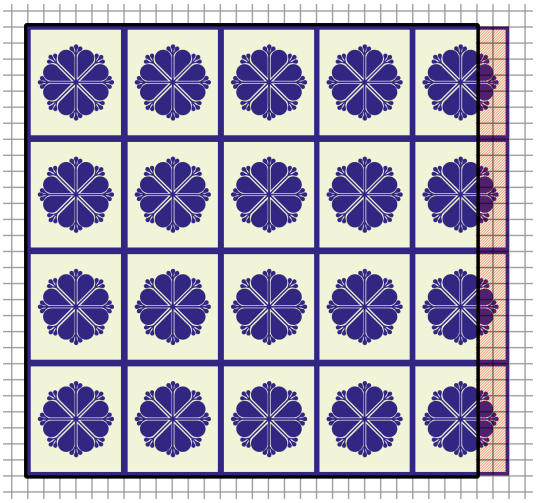
$x = 1$
The side of the small square is 1
The required area is $5 \cdot 7 = 35$



$x = 2$
The side of the small square is $\frac{1}{2}$
The required area is $4 \cdot 14 = 56$



$x = 3$
The side of the small square is $\frac{1}{3}$
The required area is $3 \cdot 7 = 21$



$x = 4$
The side of the small square is $\frac{1}{4}$
The required area is $2 \cdot 28 = 56$

Note that the size of the small square in the illustration does not affect the required total area, as the reducing laser was not applied to the unused pieces of tile.

Problem B. Exponentiator-2025

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

You are given an array $a_1, a_2, \ldots, a_n$ of positive integers. You have an exponentiator machine that can perform the following two-part operation on the array:

1. Shuffle the elements in the array $a_1, a_2, \ldots, a_n$ in any order.
2. Replace the array $a_1, a_2, \ldots, a_n$ with the array $a_1^{a_2}, a_2^{a_3}, \ldots, a_{n-1}^{a_n}$.

Thus, after each operation, the length of the array decreases by one. The exponentiator will apply $n-1$ operations, after which there will be one element left in the array a. This number is called the result of the exponentiator's work.

It is obvious that the result of the exponentiator's work is not always uniquely defined. To study the properties of the exponentiator, you need to find answers to q questions of the form: can the result of the exponentiator's work be divisible by the number x?

Input

The first line of input contains two integers n, q ($2 \le n \le 2 \cdot 10^5, 1 \le q \le 2 \cdot 10^5$) — the length of the array a and the number of queries.

The second line of input contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$) — the elements of the array a.

The next q lines contain queries, one per line, each query consists of a single number x ($1 \le x \le 10^6$).

Output

For each query, output one line: `Yes` or `No`. If the result of the exponentiator's work can be divisible by x, output `Yes`, otherwise output `No`.

Scoring

The tests for this problem consist of eight groups. Points for each group are awarded only if all tests in the group and all tests in some of the previous groups are passed.

Group	Points	Additional constraints		Required Groups	Comment
		n, q	x		
0	0	–	–	–	Tests from the statement.
1	10	$n = 2$	–	–	–
2	11	$n = 3$	–	–	–
3	12	$n = 4$	–	–	–
4	9	$n \cdot q = 10^6$	x is square-free	–	–
5	12	$n \cdot q = 10^6$	–	4	–
6	16	–	x is square-free	4	–
7	30	–	–	0-6	–

[1]A square-free number is a number that is not divisible by the square of any prime number. For example, 6 and 10 are square-free numbers, while 4 (divisible by 2^2) and 18 (divisible by 3^2) are not.

Examples

standard input	standard output
3 2 2 2 1 16 5	Yes No
7 20 1 2 3 4 5 6 7 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	Yes Yes Yes Yes Yes Yes Yes Yes No No Yes No No No Yes No Yes No No

Problem C. Bunny 3.1

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.75 seconds
Memory limit:	192 megabytes

The bunny stands in front of a staircase with n steps. The steps are numbered from 1 to n, and we can assume that the bunny is on step number 0. For better understanding, refer to the illustration in the example.

At a time, the bunny can only jump over a "beautiful" number of steps. The bunny considers a number of steps beautiful only if its representation in ternary (base 3) does not contain the digit 1. The bunny cannot jump backwards.

For example, the bunny can jump over 0 steps and thus move up one step, since $0_{10} = 0_3$ and the digit 1 was not used in the ternary representation of the number, or jump over 8 steps and move up nine steps ($8_{10} = 22_3$), but not over 7 steps ($7_{10} = 21_3$).

How many ways can the bunny reach the top (the n-th step)?

Input

The first line of input contains a single integer n ($1 \le n \le 2 \cdot 10^6$).

Output

Output a single number—the number of ways to reach the top. Since this number can be very large, output it modulo 998 244 353.

Scoring

Subtask	Points	Additional Constraints	Required Groups	Comment
0	0	–	–	Tests from the statement.
1	30	$n \le 2000$	0	–
2	15	$n \le 2 \cdot 10^4$	0–1	–
3	15	$n \le 2 \cdot 10^5$	0–2	–
4	25	$n \le 10^6$	0–3	–
5	15	—	0–4	–

Examples

standard input	standard output
7	10
5	4
10	36

Explanation

The illustration for the first example is on the next page. In this example, jumps could be made over $0_{10} = 0_3$, $2_{10} = 2_3$, and $6_{10} = 20_3$ steps.

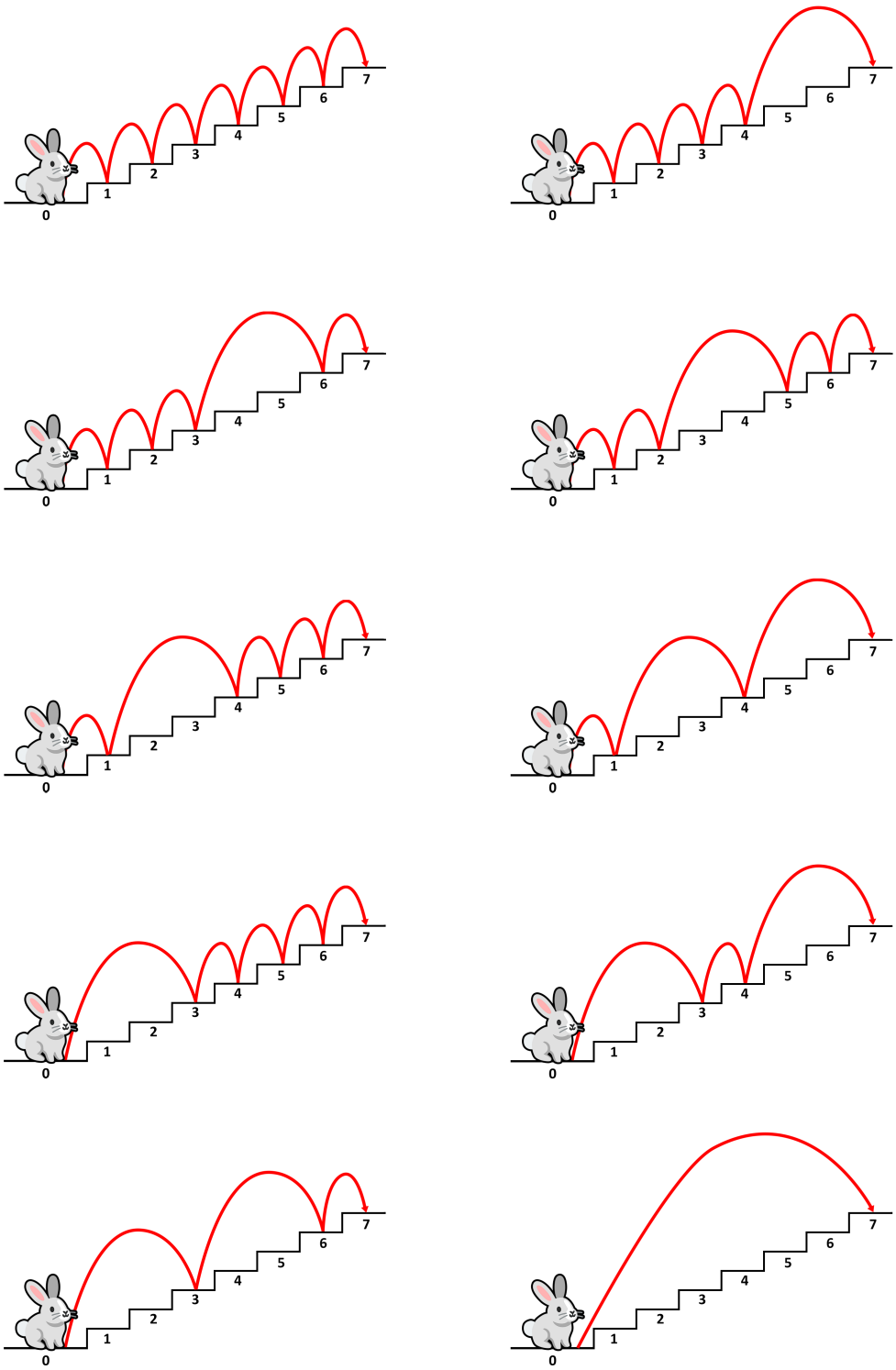


Illustration for the first example

Problem D. Metro Repair

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

The metro station has been closed for m days for repair work. You are responsible for organizing the work at the station.

There are a total of n planned tasks that can be completed within the allotted time. According to the workers' estimates, the i-th task can take any number of days from l_i to r_i. Since all tasks are technically complex and estimating their execution time is very difficult, it is guaranteed that $\frac{r_i}{l_i} \geq \alpha$ for all i from 1 to n.

Your task is to create a repair plan: select some subset of tasks to complete. To avoid being fired for inefficient allocation of labor resources, the following conditions must be met:

- You must ensure that all tasks can be completed within m days. In other words, the inequality $r_{i_1} + r_{i_2} + \ldots + r_{i_k} \leq m$ must hold, where $i_1, i_2, \ldots, i_k$ are the indices of the tasks you have chosen to complete.
- Let d be the number of days remaining until the station opens after all tasks have been completed. Then there should not exist any uncompleted j-th task that could be guaranteed to be completed within d days. Formally, for all j, the inequality $m - (l_{i_1} + l_{i_2} + \ldots + l_{i_k}) < r_j$ must hold, where $i_1, i_2, \ldots, i_k$ are the indices of the tasks you have chosen to complete, and j is the index of the task that is **not** chosen for completion.

You want to know how many subsets of tasks exist that can be selected to satisfy both of these conditions. Since the answer can be quite large, find the remainder when divided by $10^9 + 7$.

Input

The first line contains two integers n and m — the number of tasks and the total time for repair work ($1 \leq n \leq 5 \cdot 10^5$, $1 \leq m \leq 10^9$).

The second line contains a real number α — the parameter for estimating tasks ($1.5 \leq \alpha \leq 2$).

Each of the following n lines contains a pair of numbers l_i, r_i ($1 \leq l_i \leq r_i \leq m$). It is guaranteed that $\frac{r_i}{l_i} \geq \alpha$.

Output

In a single line, output the number of possible subsets of tasks to be completed modulo $10^9 + 7$.

Scoring

Group	Points	Additional Constraints		Necessary Groups	Comment
		n	α		
0	0	–	–	–	Tests from the statement.
1	12	$n \le 18$	$\alpha = 1.5$	0	–
2	13	$n \le 25$	$\alpha = 1.5$	0–1	–
3	15	$n \le 2000$	$\alpha = 2$	–	–
4	15	$n \le 2000$	$\alpha = 1.5$	0–2	–
5	10	–	$\alpha = 2$	3	–
6	16	$n \le 10^5$	$\alpha = 1.5$	0–2, 4	–
7	19	–	$\alpha = 1.5$	0–2, 4, 6	–

Examples

standard input	standard output
4 10 1.5 2 3 1 2 2 5 3 7	1
7 20 2 1 3 2 4 6 12 9 20 5 12 5 13 6 13	2

Explanation

In the first example, the only suitable subset consists of the first three tasks.

Problem E. MAX MEX MEX

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3.5 seconds
Memory limit:	256 megabytes

The hamster Ilnur gifted Alice k baskets with numbers. Each basket has a number c_i written on it.

Alice can perform the following action several times (possibly zero):

1. Choose i such that $c_i > 0$.
2. Choose exactly one number from the i-th basket.
3. Increase the chosen number from the i-th basket by 1, and decrease the number c_i by 1.

This means that in the i-th basket, this action can be performed no more than c_i times.

Alice likes the sequence $0, 1, 2, \ldots$. For each basket, she can calculate the smallest missing element needed to continue such a sequence.

More specifically, Alice can find $MEX^{\dagger}$ in each basket.

Alice became curious—what is the maximum value of MEX of all MEX that can be obtained by performing several (possibly zero) of the above actions?

Note the constraint on k and $\sum n_i$ in the last subgroup.

$^{\dagger}MEX$ (minimum excluded) of an array is the smallest non-negative integer that does not belong to the array.

For example:

MEX of the array $[2, 2, 1]$ is 0, since 0 does not belong to the array.

MEX of the array $[3, 1, 0, 1]$ is 2, since 0 and 1 belong to the array, but 2 does not.

MEX of the array $[0, 3, 1, 2]$ is 4, since 0, 1, 2, and 3 belong to the array, but 4 does not.

Input

The first line of input contains three numbers—k ($1 \le k \le 10^5$), s ($1 \le s \le 10^6$), t ($1 \le t \le 2$).

The following $2 \cdot k$ lines contain the description of the baskets with numbers.

The description of each basket depends on the value of t and looks as follows:

For $t = 1$:

The first line contains two numbers—n_i ($0 \le n_i \le s, \sum n_i = s$, i.e., the sum $n_i = s$), c_i ($0 \le c_i \le 10^{12}$).

The second line contains n_i numbers a_{i_j} ($-10^6 \le a_{i_j} \le 10^6$)—the contents of the i-th basket.

For $t = 2$:

The first line contains two numbers—n_i ($0 \le n_i \le s, \sum n_i = s$), c_i ($0 \le c_i \le 10^{12}$).

The second line contains $2 \cdot n_i$ numbers—n_i pairs of numbers a_{i_j} ($-10^6 \le a_{i_j} \le 10^6$) b_{i_j} ($1 \le b_{i_j} \le 10^6$)—the contents of the i-th basket.

Here, b_{i_j} indicates that the number a_{i_j} is repeated b_{i_j} times.

Note that if $n_i = 0$, the next line will be empty.

Output

Output a single number—$MAX(MEX(MEX))$ of all baskets with numbers after performing the necessary actions.

Scoring

The tests for this problem consist of nine groups. Points for each group are awarded only if all tests of the group and all tests of some of the previous groups are passed.

Subtask	Points	Additional Constraints	Required Groups	Comment
0	0	–	–	Tests from the statement.
1	10	$t = 1, n_i \le 15, k \le 1000$	–	–
2	10	$t = 1, c_i = 0$	–	–
3	15	$t = 1, a_{i_j}$ are distinct in one basket	–	–
4	35	$t = 1$	1–3	–
5	30	$t = 2, k \le 1000, s \le 3 \cdot 10^4$	–	–

Examples

standard input	standard output
4 5 2 1 5 -4 1 1 0 -1000000 1000000000 1 3 0 3 2 100 10 3 -4 2	4
4 8 1 1 5 -4 0 10 3 3 0 0 0 4 1000000000000 100 -1000000 -1 957	4

Note

In the first set of input data, Alice can perform the following actions:

Basket 1 $[-4] \Rightarrow [-3] \Rightarrow [-2] \Rightarrow [-1] \Rightarrow [0]$.

Basket 2 is empty $[]$.

Basket 3 $[0, 0, 0] \Rightarrow [0, 0, 1] \Rightarrow [0, 1, 1] \Rightarrow [0, 1, 2]$

Basket 4 $[100, -1000000, -1, 957] \Rightarrow \ldots \Rightarrow [100, 0, 1, 957]$

Then $MEX_1 = 1, MEX_2 = 0, MEX_3 = 3, MEX_4 = 2$.

Thus, $MEX([MEX_1, MEX_2, MEX_3, MEX_4]) = MEX([1, 0, 3, 2]) = 4$.

In the second set of input data, Alice can perform the following actions:

Basket 1 $[-4] \Rightarrow [-3] \Rightarrow [-2] \Rightarrow [-1] \Rightarrow [0]$.

Basket 2 $[-1000000, -1000000, \ldots, -1000000]$ will remain unchanged.

Basket 3 $[0, 0, 0] \Rightarrow [0, 0, 1] \Rightarrow [0, 1, 1] \Rightarrow [0, 1, 2]$

Basket 4 $[10, 10, 10, -4, -4] \Rightarrow \ldots \Rightarrow [10, 10, 10, 0, 1]$

Then $MEX_1 = 1, MEX_2 = 0, MEX_3 = 3, MEX_4 = 2$.

Thus, $MEX([MEX_1, MEX_2, MEX_3, MEX_4]) = MEX([1, 0, 3, 2]) = 4$.