Codeforces Round #330 (Div. 2)

A. Vitaly and Night

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

One day Vitaly was going home late at night and wondering: how many people aren't sleeping at that moment? To estimate, Vitaly decided to look which windows are lit in the house he was passing by at that moment.

Vitaly sees a building of n floors and $2 \cdot m$ windows on each floor. On each floor there are m flats numbered from 1 to m, and two consecutive windows correspond to each flat. If we number the windows from 1 to $2 \cdot m$ from left to right, then the j-th flat of the i-th floor has windows $2 \cdot j$ - 1 and $2 \cdot j$ in the corresponding row of windows (as usual, floors are enumerated from the bottom). Vitaly thinks that people in the flat aren't sleeping at that moment if **at least one** of the windows corresponding to this flat has lights on.

Given the information about the windows of the given house, your task is to calculate the number of flats where, according to Vitaly, people aren't sleeping.

Input

The first line of the input contains two integers n and m ($1 \leq n, m \leq 100$) — the number of floors in the house and the number of flats on each floor respectively.

Next n lines describe the floors from top to bottom and contain $2 \cdot m$ characters each. If the i-th window of the given floor has lights on, then the i-th character of this line is '1', otherwise it is '0'.

Output

Print a single integer — the number of flats that have lights on in at least one window, that is, the flats where, according to Vitaly, people aren't sleeping.

Sample test(s)

input
2 2 0 0 0 1 1 0 1 1
output
3

input
1 3 1 1 0 1 0 0
output
2

Note

In the first test case the house has two floors, two flats on each floor. That is, in total there are 4 flats. The light isn't on only on the second floor in the left flat. That is, in both rooms of the flat the light is off.

In the second test case the house has one floor and the first floor has three flats. The light is on in the leftmost flat (in both windows) and in the middle flat (in one window). In the right flat the light is off.

B. Pasha and Phone

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

Pasha has recently bought a new phone `jPager` and started adding his friends' phone numbers there. Each phone number consists of exactly n digits.

Also Pasha has a number k and two sequences of length n / k (n is divisible by k) $a_1, a_2, ..., a_{n/k}$ and $b_1, b_2, ..., b_{n/k}$. Let's split the phone number into blocks of length k. The first block will be formed by digits from the phone number that are on positions $1, 2,..., k$, the second block will be formed by digits from the phone number that are on positions $k + 1, k + 2, ..., 2 \cdot k$ and so on. Pasha considers a phone number `good`, if the i-th block doesn't start from the digit b_i and is divisible by a_i if represented as an integer.

To represent the block of length k as an integer, let's write it out as a sequence $c_1, c_2,...,c_k$. Then the integer is calculated as the result of the expression $c_1 \cdot 10^{k-1} + c_2 \cdot 10^{k-2} + ... + c_k$.

Pasha asks you to calculate the number of `good` phone numbers of length n, for the given k, a_i and b_i. As this number can be too big, print it modulo $10^9 + 7$.

Input

The first line of the input contains two integers n and k ($1 \le n \le 100\,000$, $1 \le k \le min(n, 9)$) — the length of all phone numbers and the length of each block, respectively. It is guaranteed that n is divisible by k.

The second line of the input contains n / k space-separated positive integers — sequence $a_1, a_2, ..., a_{n/k}$ ($1 \le a_i < 10^k$).

The third line of the input contains n / k space-separated positive integers — sequence $b_1, b_2, ..., b_{n/k}$ ($0 \le b_i \le 9$).

Output

Print a single integer — the number of good phone numbers of length n modulo $10^9 + 7$.

Sample test(s)

input
6 2 38 56 49 7 3 4
output
8

input
8 2 1 22 3 44 5 4 3 2
output
32400

Note

In the first test sample `good` phone numbers are: `000000`, `000098`, `005600`, `005698`, `380000`, `380098`, `385600`, `385698`.

C. Warrior and Archer

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

In the official contest this problem has a different statement, for which jury's solution was working incorrectly, and for this reason it was excluded from the contest. This mistake have been fixed and the current given problem statement and model solution corresponds to what jury wanted it to be during the contest.

Vova and Lesha are friends. They often meet at Vova's place and compete against each other in a computer game named The Ancient Papyri: Swordsink. Vova always chooses a warrior as his fighter and Leshac chooses an archer. After that they should choose initial positions for their characters and start the fight. A warrior is good at melee combat, so Vova will try to make the distance between fighters as small as possible. An archer prefers to keep the enemy at a distance, so Lesha will try to make the initial distance as large as possible.

There are n (n is always even) possible starting positions for characters marked along the Ox axis. The positions are given by their distinct coordinates $x_1, x_2, ..., x_n$, two characters cannot end up at the same position.

Vova and Lesha take turns banning available positions, Vova moves first. During each turn one of the guys bans exactly one of the remaining positions. Banned positions cannot be used by **both** Vova and Lesha. They continue to make moves until there are only two possible positions remaining (thus, the total number of moves will be $n - 2$). After that Vova's character takes the position with the lesser coordinate and Lesha's character takes the position with the bigger coordinate and the guys start fighting.

Vova and Lesha are already tired by the game of choosing positions, as they need to play it before every fight, so they asked you (the developer of the The Ancient Papyri: Swordsink) to write a module that would automatically determine the distance at which the warrior and the archer will start fighting if both Vova and Lesha play optimally.

Input

The first line on the input contains a single integer n ($2 \le n \le 200\,000$, n is even) — the number of positions available initially. The second line contains n distinct integers $x_1, x_2, ..., x_n$ ($0 \le x_i \le 10^9$), giving the coordinates of the corresponding positions.

Output

Print the distance between the warrior and the archer at the beginning of the fight, provided that both Vova and Lesha play optimally.

Sample test(s)

input
6 0 1 3 7 15 31
output
7

input
2 73 37
output
36

Note

In the first sample one of the optimum behavior of the players looks like that:

1. Vova bans the position at coordinate 15;
2. Lesha bans the position at coordinate 3;
3. Vova bans the position at coordinate 31;
4. Lesha bans the position at coordinate 1.

After these actions only positions 0 and 7 will remain, and the distance between them is equal to 7.

In the second sample there are only two possible positions, so there will be no bans.

D. Max and Bike

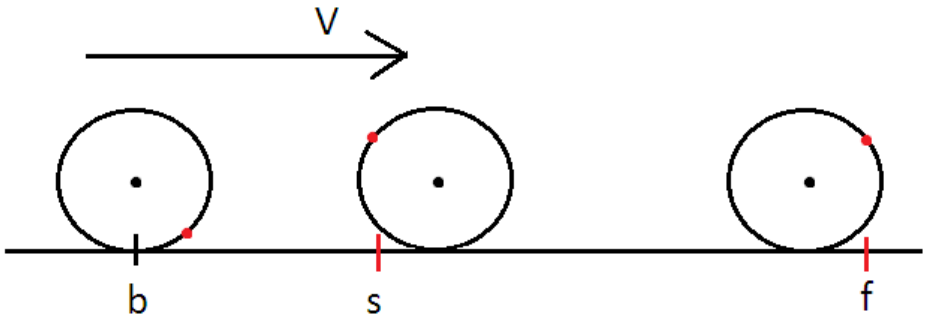
time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

For months Maxim has been coming to work on his favorite bicycle. And quite recently he decided that he is ready to take part in a cyclists' competitions.

He knows that this year n competitions will take place. During the i-th competition the participant must as quickly as possible complete a ride along a straight line from point s_i to point f_i ($s_i < f_i$).

Measuring time is a complex process related to usage of a special sensor and a time counter. Think of the front wheel of a bicycle as a circle of radius r. Let's neglect the thickness of a tire, the size of the sensor, and all physical effects. The sensor is placed on the rim of the wheel, that is, on some fixed point on a circle of radius r. After that the counter moves just like the chosen point of the circle, i.e. moves forward and rotates around the center of the circle.

At the beginning each participant can choose **any** point b_i, such that his bike is fully behind the starting line, that is, $b_i < s_i - r$. After that, he starts the movement, instantly accelerates to his maximum speed and at time ts_i, when the coordinate of the sensor is equal to the coordinate of the start, the time counter starts. The cyclist makes a complete ride, moving with his maximum speed and at the moment the sensor's coordinate is equal to the coordinate of the finish (moment of time tf_i), the time counter deactivates and records the final time. Thus, the counter records that the participant made a complete ride in time $tf_i - ts_i$.



Maxim is good at math and he suspects that the total result doesn't only depend on his maximum speed v, but also on his choice of the initial point b_i. Now Maxim is asking you to calculate for each of n competitions the minimum possible time that can be measured by the time counter. The radius of the wheel of his bike is equal to r.

Input

The first line contains three integers n, r and v ($1 \le n \le 100\,000$, $1 \le r, v \le 10^9$) — the number of competitions, the radius of the front wheel of Max's bike and his maximum speed, respectively.

Next n lines contain the descriptions of the contests. The i-th line contains two integers s_i and f_i ($1 \le s_i < f_i \le 10^9$) — the coordinate of the start and the coordinate of the finish on the i-th competition.

Output

Print n real numbers, the i-th number should be equal to the minimum possible time measured by the time counter. Your answer will be considered correct if its absolute or relative error will not exceed 10^{-6}.

Namely: let's assume that your answer equals a, and the answer of the jury is b. The checker program will consider your answer correct if $\frac{|a-b|}{\max(1,b)} \le 10^{-6}$.

Sample test(s)

input
2 1 2 1 10 5 9
output
3.849644710502 1.106060157705

E. Edo and Magnets

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

Edo has got a collection of n refrigerator magnets!

He decided to buy a refrigerator and hang the magnets on the door. The shop can make the refrigerator with any size of the door that meets the following restrictions: the refrigerator door must be rectangle, and both the length and the width of the door must be **positive integers**.

Edo figured out how he wants to place the magnets on the refrigerator. He introduced a system of coordinates on the plane, where each magnet is represented as a rectangle with sides parallel to the coordinate axes.

Now he wants to remove no more than k magnets (he may choose to keep all of them) and attach all remaining magnets to the refrigerator door, and the area of the door should be as small as possible. A magnet is considered to be attached to the refrigerator door if **its center** lies on the door or on its boundary. The relative positions of all the remaining magnets must correspond to the plan.

Let us explain the last two sentences. Let's suppose we want to hang two magnets on the refrigerator. If the magnet in the plan has coordinates of the lower left corner (x_1, y_1) and the upper right corner (x_2, y_2), then its center is located at $(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2})$ (may not be integers). By saying the relative position should correspond to the plan we mean that the only available operation is translation, i.e. the vector connecting the centers of two magnets in the original plan, must be equal to the vector connecting the centers of these two magnets on the refrigerator.

The sides of the refrigerator door must also be parallel to coordinate axes.

Input

The first line contains two integers n and k $(1 \le n \le 100\,000, 0 \le k \le min(10, n - 1))$ — the number of magnets that Edo has and the maximum number of magnets Edo may not place on the refrigerator.

Next n lines describe the initial plan of placing magnets. Each line contains four integers x_1, y_1, x_2, y_2 $(1 \le x_1 < x_2 \le 10^9, 1 \le y_1 < y_2 \le 10^9)$ — the coordinates of the lower left and upper right corners of the current magnet. The magnets can partially overlap or even fully coincide.

Output

Print a single integer — the minimum area of the door of refrigerator, which can be used to place at least $n - k$ magnets, preserving the relative positions.

Sample test(s)

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

input
4 1 1 1 2 2 1 9 2 10 9 9 10 10 9 1 10 2
output
64

input
3 0 1 1 2 2 1 1 1000000000 1000000000 1 3 8 12
output
249999999000000001

Note

In the first test sample it is optimal to remove either the first or the third magnet. If we remove the first magnet, the centers of two others will lie at points (2.5, 2.5) and (3.5, 3.5). Thus, it is enough to buy a fridge with door width 1 and door height 1, the area of the door also equals one, correspondingly.

In the second test sample it doesn't matter which magnet to remove, the answer will not change — we need a fridge with door width 8 and door height 8.

In the third sample you cannot remove anything as $k = 0$.