Analysis

In order to be able to solve the problem we have to learn to find the exact moment in which two particles collide. Let us try to find the moment in which the *i*-th *x* particle and the *j*-th *y* particle collide. We can do that quite easily using a binary search on the moment of collision, but that would have a complexity of $O(log)$. To find the moment of collision in $O(1)$ let us look at the formulas that describe the position of a certain particle in a specific moment. Let us imagine that the *x* particles are being shot from point 0 rightwards on the number line, and the *y* particles are being shot from point L leftwards on the number line. At a certain moment *t*, the position of the two particles is:

$Px_i = (t - tx_i) \times vx_i$

$Py_j = L - (t - ty_j) \times vy_j$

This formula could give negative values if we evaluate it for invalid times in which the particle has not been shot yet, but that does not matter. We clearly have a collision if $Px_i = Py_j$. That is:

$$(t - tx_i) \times vx_i = L - (t - ty_j) \times vy_j$$
$$t \times vx_i - tx_i \times vx_i = L - t \times vy_j + ty_j \times vy_j$$
$$t \times vx_i + t \times vy_j = L + ty_j \times vy_j + tx_i \times vx_i$$

$$t = \frac{L - ty_j \times vy_j + tx_i \times vx_i}{vx_i + vy_j}$$

And hence we get a direct formula for the moment of collision. It is possible for the collision to happen before 0 or after L on the number line, which means that this collision will surely not happen (those particles will collide with some others before colliding with each other).

Solution with complexity $O(N^2 \times logN)$

Since we can quickly find the moment of collision between two particles, we can find the moment of collision for every pair of particles and order these collisions chronologically. The total amount of collisions is $O(N^2)$, and sorting them takes $O(N^2 \times logN)$. After this we can iterate over all collisions chronologically and keep which of the particles have already collided. If we reach a pair in which both particles have not collided with anything yet, then we know that they will collide with each other. In this way we can find not only the first K, but all collisions with the same complexity.

Solution with complexity $O(N \times K \times log)$

We will describe a solution that finds only the first collision. After finding it we can remove the two particles that collided and repeat the whole process K times.

To find the first collision we will use binary search on the moment of the first collision. Let us fix the time *t*. We want to know whether the first collision happened before or after *t*. To do this we can find the furthest *x* particle and the furthest *y* particle (by 'furthest' we mean the

one that travelled the largest distance, that is, using the formulas above, the *x* particle with the largest position and the *y* particle with the smallest one). Let the positions of the two particles, according to the formulas above, be respectively P_x and P_y. Then:

- If $|P_x - P_y| < \varepsilon$ (where ε is a small enough constant) we can assume that *t* is the moment of the first collision. The particles which collided are the furthest particles..

Else:

- If $P_x < P_y$, then the first collision has not happened yet and we have to look for a larger value of *t*
- If $P_x > P_y$, then the first collision already happened and we have to look for a smaller value of *t*

This solution has a complexity of $O(N \times log)$ for finding the first collision. The bounds of the binary search in which we search for *t* are from 0 to the time needed for any particle to be shot and travel a distance of *L* (since at this moment the particle must have collided with some other one).
Since we are doing a binary search on real values and not integers, it is a good idea to set a limit on the amount of iterations the binary search can perform, since choosing a good ε may be tricky. About 40-50 iterations should be ideal (more may trigger time limits and less could lead to precision errors).
Using this procedure K times we get a solution with total complexity of $O(N \times K \times log)$.

The problem can be solved in $O(N \times K)$, but this was not required.