KOLICA

Simulating second by second will be too slow because the coordinates of the carts are too large.

The largest number of potential collisions we need to consider is on the order of N^2, when we consider every pair of carts. The largest number of actual collisions is on the order of N, because at least two carts disappear in every collision. One solution is to, up to N times, search for the earliest next collision, and then simulates the movement of all carts up to that moment. Such a solution has complexity $O(N^3)$.

For two carts to collide, it is necessary that their distance in half a second (or less) is smaller than it is now. If this condition is satisfied, then the carts will really collide if they are moving towards one another, or perpendicularly with $abs(x_1-x_2) = abs(y_1-y_2)$.

Another possible solution is to find all possible collisions, sort them in ascending order by time and then simulate event by event, where a potential collision really occurs if all carts taking part are still alive. The complexity of such a solution is $O(N^2 \log N)$ when using an appropriate sorting algorithm.

POTPUNI

Because the stored number can become very large, we need to notice that a number is a perfect square only if, when factorized into prime factors, each of its prime factors appears an even number of time.

We need to keep track of how many times each prime factor occurs in the factorization, and how many factors appear an odd number of times. The stored number is a perfect square if the second of these numbers is zero.

As we input each number, we need to factorize it, and refresh the table saying how many times each prime factor occurs.

To factorize each number, it suffices to know any of its prime factors (for example, the smallest). We preprocess this factor for each possible number with a simple modification of the sieve of Eratosthenes. The prime factorization of a number is then found by repeatedly dividing the number with any of its prime factors.

OGRADA

For Lea to earn the largest amount, she can distribute the boards by the following rules, considering boards from most expensive to cheapest:

1. If possible, put the board in front of the highest uncovered board in Slavko's fence which is not higher than Lea's.
2. If such a board doesn't exist, Lea needs to put her board in front of Slavko's highest uncovered board.

One possible implementation is to sort Lea's boards in descending order by price and then search through all of Slavko's boards. Such an implementation has time complexity $O(N^2)$, which is too slow.

To avoid repeatedly searching through all of Slavko's boards, we need to use a data structure which allows for efficient erasing and binary searching. One such data structure is a binary search tree, for an overall time complexity of $O(N \log N)$.