As an employee of the world's most respected political polling corporation, you must take complex, realworld issues and simplify them down to a few numbers. It isn't always easy. A big election is coming up and, at the request of Candidate X, you have just finished polling n people. You have gathered three pieces of information from each person, with the values for the i-th person recorded as:

- a_i — the number of digits of π they have memorized
- b_i — the number of hairs on their head
- c_i — whether they will vote for Candidate X

Unfortunately, you are beginning to wonder if these are really the most relevant questions to ask. In fact, you cannot see any correlation between a, b, and c in the data. Of course, you cannot just contradict your customer — that is a good way to lose your job!

Perhaps the answer is to find some weighting formula to make the results look meaningful. You will pick two real values S and T, and sort the poll results (a_i, b_i, c_i) by the measure $a_i \cdot S + b_i \cdot T$. The sort will look best if the results having c_i true are clustered as close to each other as possible. More precisely, if j and k are the indices of the first and last results with c_i true, you want to minimize the cluster size which is $k - j + 1$. Note that some choices of S and T will result in ties among the (a_i, b_i, c_i) triples. When this happens, you should assume the worst possible ordering occurs (that which maximizes the cluster size for this (S, T) pair).

Input

The input file contains several test cases, each of them as described below.

The input starts with a line containing n ($1 \leq n \leq 250000$), which is the number of people polled. This is followed by one line for each person polled. Each of those lines contains integers a_i ($0 \leq a_i \leq 2000000$), b_i ($0 \leq bi \leq 2000000$), and c_i, where c_i is '1' if the person will vote for Candidate X and '0' otherwise. The input is guaranteed to contain at least one person who will vote for Candidate X.

Output

For each test case, display the smallest possible cluster size over all possible (S, T) pairs.

Sample Input

```
6
0 10 0
10 0 1
12 8 1
5 5 0
11 2 1
11 3 0
10
6 1 1
0 2 0
2 1 1
6 1 1
8 2 0
4 4 0
4 0 0
2 3 1
6 1 0
6 3 1
```

Sample Output

```
4
8
```