ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Algorithms Lab HS17
Department of Computer Science
Prof. Dr. A. Steger, Prof. Dr. E. Welzl, Prof. Dr. P. Widmayer
cadmo.ethz.ch/education/lectures/HS17/algolab

Threefold Problem Sheet — Moretorcycles

This sheet presents three problems based on a common storyline. The three problems share several characteristics and as a result they appear quite similar in nature. But this does not necessarily mean that these problems can be solved using the same strategies and techniques. In fact, the whole idea of these threefold sheets is to highlight how possibly subtle differences and changes in a problem formulation can have a great impact on how the resulting problem can be solved. We believe that studying these sheets is an excellent practice of how to approach the modeling aspects of solving a problem. For these problems you do not have to write code and there will be no testsets and no corresponding problem on the judge.

Read through the problems and try to work out by yourself possible lines of modeling and then algorithmically solving them. Give reasons of why or why not certain approaches appear plausible. Ideally, substantiate your ideas by sketching corresponding proof ideas.

This sheet was handed out, solved and discussed during the tutorial on November 22, 2017.

Main Story

Surely you fondly remember our 5^{th} Problem of the Week, called "Motorcycles", from October 23, 2017. Let us recapitulate the problem statement.

It is an old biker's dream to ride forever into the sunrise. Alas, not everybody manages to live the dream. Who will drive on forever? Consider n bikers that start on the y-axis and ride east, into the positive x-halfplane. All bikers start at the same time and drive with the same speed. Every biker follows a straight path. However, if a biker runs into the tracks of another biker, she loses the desire to continue on the trip and stops right there. (The point of riding into the sunrise is to boldly go where no ... well, at least not follow someone else's tracks.)

Your job was to compute and output the indices of all bikers that ride forever into the sunrise, which is defined as follows. A biker *rides forever into the sunrise* if she never meets the tracks of any other biker (that is, a point where another biker has been to earlier than her). In case two bikers reach the same spot at exactly the same time, by the usual traffic regulations the biker that comes from the right takes precedence and continues her drive; the biker that came from the left ends her journey there.

Here we pick up this story and vary the problem setting in three different ways.

Exercise – *Admission Control*

It is frustrating to see so many bikers give up along the way. Maybe there are just too many bikers, so that they get into each other's way too easily? In this mindset, the Sunriding Vehicle Party started lobbying for an initiative "Against Massbiking" so as to restrict the number of participants of the annual sunriding event. You are charged to investigate the impact of such a restriction.

Given a collection of bikers, you can select who is allowed to participate and who is excluded from participation. Your goal is to make this selection so that

1. all selected participants ride forever into the sunrise *and*
2. the number of selected participants is maximized (equivalently, the number of bikers excluded from participation is minimized).

Input The first line of the input contains the number $t \leq 30$ of test cases. Each of the t test cases is described as follows.

- It starts with a line that contains a single integer n so that $1 \leq n \leq 5 \cdot 10^5$. Here n denotes the number of bikers.
- The following n lines define the starting positions and driving directions of the bikers $b_0, \ldots, b_{n-1}$. Each line contains three integers `y0 x1 y1` separated by a space and such that $x_1 > 0$ and $|y_0|, x_1, |y_1| < 2^{51}$. The corresponding biker starts at position $(0, y_0)$ and rides off from there in direction of the point (x_1, y_1). Note that the point (x_1, y_1) specifies the direction not the destination. That is, the biker does not stop if she reaches this point.

You may assume that the starting positions are pairwise distinct.

Output For each test case output a single line that consists of two integers a and b, separated by a space. Here

- a denotes the number of bikers that ride forever into the sunrise if all bikers participate *and*
- b denotes the maximum number of participants that can be allowed so that all participants ride forever into the sunrise.

Sample Input

```
2
4
3 1 3
2 1 3
1 2 2
0 3 1
3
1 1 0
2 1 0
-1 2 0
```

Sample Output

```
1 3
1 1
```

Exercise – *Starting Schedules*

The arguments in favor of an admission control appear convincing at first glance: The number of bikers that ride forever into the sunrise is never smaller than without such a control, and it is significantly higher in some cases. However, the number of participants may also drop dramatically. In an extreme case, out of millions of bikers only a single one would be allowed to participate. What a ridiculous regime! There has to be a better way...

For once, the bikers should drop their attitude and not give up so easily. It is about the journey, not about the destination! We should probably temper our expectations concerning the biker's frustration tolerance. But it seems reasonable to expect a biker to continue when hitting the tracks of a biker that passed by only recently ("Ah, they're not far ahead...").

Now it is your job to make the best of the time windows in which the bikers can cross each other's track. To this end you assign to every biker b_i, $i \in \{0, \ldots, n-1\}$, an individual starting time $s_i \in \mathbb{R}$, $s_i \geq 0$, to begin the journey. We assume that all bikers travel at the same unit speed, covering a distance of one in one unit of time.

For simplicity assume that all bikers have the same (unknown) frustration tolerance $f \geq 0$. Whenever a biker b_i meets the tracks of another biker b_j, the following happens: If b_j passed this place no more than f time units earlier, then b_i continues; otherwise, b_i stops.

A biker that never stops *rides forever into the sunrise*. What is the minimum frustration tolerance $f \geq 0$ required so that a suitable starting schedule allows all bikers to ride forever into the sunrise?

Input The first line of the input contains the number $t \leq 30$ of test cases. Each of the t test cases is described as follows.

- It starts with a line that contains a single integer n so that $1 \leq n \leq 10^2$. Here n denotes the number of bikers.
- The following n lines define the starting positions and driving directions of the bikers $b_0, \ldots, b_{n-1}$. Each line contains three integers y_0 x_1 y_1 separated by a space and such that $x_1 > 0$ and $|y_0|, x_1, |y_1| < 2^{51}$. The corresponding biker starts at position $(0, y_0)$ and rides off from there in direction of the point (x_1, y_1).

You may assume that the starting positions are pairwise distinct.

Output For each test case output a single line that consists of a single integer f that denotes the minimum integral frustration tolerance required so that there exists a starting schedule that allows all bikers to ride forever into the sunrise.

Sample Input

```
2
3
3 4 0
1 1 2
0 1 0
3
6 8 0
0 1 0
-1 1 0
```

Sample Output

```
0
2
```

Exercise – *Beethoven*

It is well-known that bikers enjoy listening to classical music while riding into the sunrise. Therefore, a number of radio stations have been set up to broadcast string quartets and piano sonatas as on-demand audio along the way. Eventually, the bikers may leave the station's coverage. But at least for an initial strip $S_w = [0, w] \times \mathbb{R} \subset \mathbb{R}^2$ of width w radio reception is to be guaranteed.

Each radio signal can be received within a radius of $r \geq 0$ around its station's position. For a given width $w \geq 0$, the track of a biker is *covered* if at least one radio station can be received at any point in S_w that the biker passes through during her journey. How large does r have to be so as to cover the tracks of all bikers?

The music has the desirable side effect that the bikers no longer care—in fact, they are too distracted to notice—if they run into the tracks of another biker. Using the terminology of the previous exercise, their frustration tolerance is ∞.

Input The first line of the input contains the number $t \leq 30$ of test cases. Each of the t test cases is described as follows.

- It starts with a line that contains three integers `n m w`, separated by a space. They denote
  - n, the number of bikers ($1 \leq n \leq 3 \cdot 10^3$);
  - m, the number of antennas ($1 \leq m \leq 3 \cdot 10^3$);
  - w, the width of the strip ($0 \leq w \leq 2^{51}$).
- The following n lines define the starting positions and driving directions of the bikers $b_0, \ldots, b_{n-1}$. Each line contains three integers y_0 x_1 y_1 separated by a space and such that $x_1 > 0$ and $|y_0|, x_1, |y_1| < 2^{51}$. The corresponding biker starts at position $(0, y_0)$ and rides off from there in direction of the point (x_1, y_1).
- The following m lines define the positions of the antennas. Each line contains two integers `x y` separated by a space and such that $x > 0$ and $x, |y| < 2^{51}$.

You may assume that the starting positions of the bikers are pairwise distinct.

Output For each test case output a single line that consists of a single integer `r` that denotes the minimum integral broadcast radius necessary to cover all biker's tracks.

Sample Input

```
1
3 2 5
3 4 0
1 1 2
0 1 0
2 1
7 3
```

Sample Output

```
4
```

Briefly sketch your approach for each of the three exercises in keywords.

Exercise 1 – *Admission Control*

Method(s): ☐ Brute force ☐ Greedy ☐ Backtracking ☐ Dynamic programming ☐ Sorting ☐ Sliding window ☐ Binary search ☐ Graph components ☐ Minimum spanning tree ☐ BFS/DFS ☐ Shortest paths (Dijkstra) ☐ Maximum matching ☐ Network flows ☐ MinCost flows ☐ Delaunay triangulation ☐ Minimum enclosing circle ☐ LP ☐ QP

Time Complexity:

Reasoning:

Exercise 2 – *Starting Schedules*

Method(s): ☐ Brute force ☐ Greedy ☐ Backtracking ☐ Dynamic programming ☐ Sorting ☐ Sliding window ☐ Binary search ☐ Graph components ☐ Minimum spanning tree ☐ BFS/DFS ☐ Shortest paths (Dijkstra) ☐ Maximum matching ☐ Network flows ☐ MinCost flows ☐ Delaunay triangulation ☐ Minimum enclosing circle ☐ LP ☐ QP

Time Complexity:

Reasoning:

Exercise 3 – *Beethoven*

Method(s): ☐ Brute force ☐ Greedy ☐ Backtracking ☐ Dynamic programming ☐ Sorting ☐ Sliding window ☐ Binary search ☐ Graph components ☐ Minimum spanning tree ☐ BFS/DFS ☐ Shortest paths (Dijkstra) ☐ Maximum matching ☐ Network flows ☐ MinCost flows ☐ Delaunay triangulation ☐ Minimum enclosing circle ☐ LP ☐ QP

Time Complexity:

Reasoning: