Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter

**Central Europe Regional Contest 2018**

# Incredible Hull

`hull.c`, `hull.cpp`, `Hull.java`, `hull.py`

In a royal casino, there is a huge room with many slot machines. The casino manager feels that this room may be too easy to navigate and so the people have a good chance of getting out of the room without losing much money. This needs to be fixed, therefore the manager designed a complicated system of slot machine positions and straight aisles between them, so people get lost there more likely.

The room's shape is roughly oval. If it was empty, the whole room could be observed from any inside point. Each of $N$ slot machines in the room can be represented as a point and it is assigned some unique profit value. The manager wants the slot machines to be connected by aisles in the way described below.

There already are at least 3 machines built into the wall of the room. All these wall machines must be connected by straight aisles to create a closed perimeter path around the whole room. All of the remaining slot machines in the room (if there are any) will be inside the area enclosed by the perimeter path.

The room is to be divided by straight aisles into smaller separate areas, each enclosed by its own perimeter path. The division proceeds in two phases.

- The first phase is ruled by the following scheme. If there are at least 4 slot machines on a perimeter path, two pivot machines have to be chosen. The first pivot machine is the most profitable one on the perimeter path. The second pivot machine is also on the perimeter path and it is the furthest one from the first one. The distance between machines is measured along the perimeter path and it is equal to the number of aisles one has to traverse to get from one machine to the other one. If there are two furthest machines, the more profitable one has to be selected as the second pivot machine. The area inside the perimeter path is divided into two areas by a new aisle which connects the pivot machines. The perimeter path of each of the two new smaller areas is the minimal part of the original perimeter path which together with the new aisle form a closed path encircling the smaller area. The scheme is applied repeatedly until no new area can be created.
- When the previous division phase is completed, the second division phase begins. It is ruled by the following scheme. For any area $A$ with no aisle inside it, consider all slot machines inside that area. If there are any, connect the most profitable one of them by aisles to all slot machines on the perimeter path $P_A$ of the area $A$. This divides the area into several smaller areas. The perimeter path of each new area consists of one aisle of $P_A$ and two newly created aisles. The area lies inside the triangle formed by these three aisles and it contains no other aisles. The scheme is applied repeatedly until no new area can be created.

The Manager wants to assess the profitability of his design. For this purpose, he defines a so-called *highly-profitable configuration*, which is a maximum-size set of machines with a property that each pair of machines in the set is directly connected by an aisle.

The manager is specifically interested in three *profitability parameters*. The first profitability parameter is the size of one highly-profitable configuration. The second profitability parameter is the number of highly-profitable configurations that exist in the design. The third profitability parameter is the number of slot machines which are part of at least one highly-profitable configuration.

## Input Specification

The first line of input contains a single integer $N$ ($3 \leq N \leq 10^5$), the number of slot machines. Each of the next $N$ lines contains two integers $X$, $Y$ ($0 \leq X, Y \leq 10^9$), denoting the position of one slot machine in the room. The machines are listed in decreasing order of their profit value. The positions of no three machines are collinear.

## Output Specification

Output the three profitability parameters.

### Sample Input 1

```
6
8 2
6 8
4 9
3 5
3 0
1 5
```

### Output for Sample Input 1

```
4 1 4
```

### Sample Input 2

```
6
1 1
6 1
1 6
6 6
3 2
4 5
```

### Output for Sample Input 2

```
4 2 6
```