Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

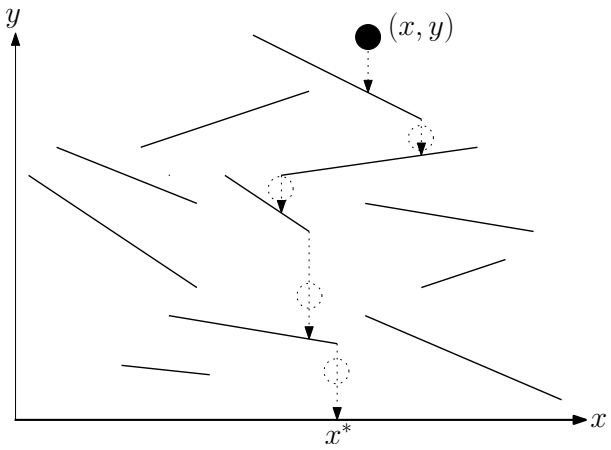
Institut für Theoretische Informatik
Peter Widmayer
Thomas Tschager
Antonis Thomas

1st June 2016

Datenstrukturen & Algorithmen Exercise Sheet 14 FS 16

Exercise 14.1 *Rolling Balls.*

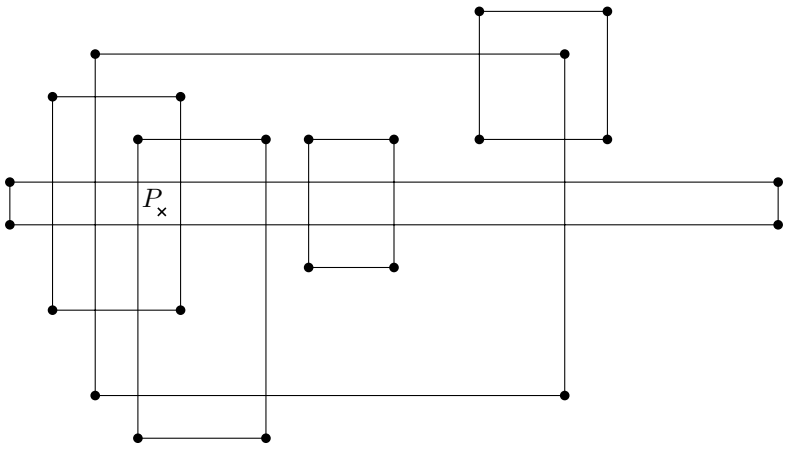
We consider a ball rolling down an arrangement of timber channels. The ball is released at a given position and falls straight down until it hits the first channel. Then, it rolls down along this channel and falls straight down at the lower end of the channel. This repeats until the ball hits the ground. We represent the channels by line segments and assume that no two segments are crossing or touching each other. The x and y coordinates of the endpoints are pairwise different. Moreover, no segment is horizontal or vertical. The starting position of the ball is given by a coordinate (x, y).



Describe an algorithm that computes efficiently on which position x^* the ball hits the ground. Which running time in dependency of the number of line segments does this algorithm have?

Exercise 14.2 *Piercing Orthogonal Rectangles.*

You are given n orthogonal rectangles which may overlap. We are searching for a point that maximizes the number of pierced rectangles. A rectangle is pierced by a point if and only if it is lying inside or on the boundary. In the following example, P is such a point because it pierces four rectangles while no other point exists (outside the direct environment of P) that pierces more.



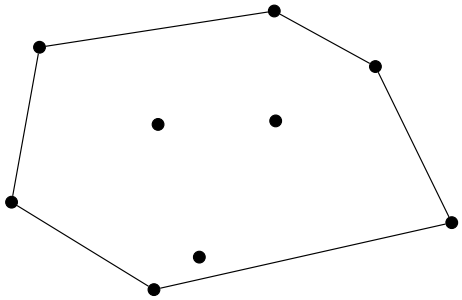
Describe an efficient scanline algorithm to find a point that maximizes the number of pierced rectangles. Provide also the running time of your solution.

Exercise 14.3 *B-Trees.*

Give an example of a B-tree with order 5 and height 3 and an additional new key, such that the insertion of this new key increases the height of the tree.

Exercise 14.4 *Convex Hull (Programming exercise).*

In this exercise we are going to implement Graham's Scan; an algorithm that computes the convex hull of a given point set. Let $p_1, ..., p_n \in \mathbb{N}^2$ be points in the plane with integer coordinates and in general position (i.e., no three of them lie on a straight line). The goal is to compute the *convex hull*, defined as the smallest convex set containing all the n points. The following picture shows an example of a convex hull of a point set.



Input The first line contains only the number t of test instances. After that, we have exactly one line per test instance. Every line contains the sequence $n, x_1, y_1, ..., x_n, y_n$. The number $n \in \mathbb{N}$, $3 \leq n \leq 1000$, describes the number of points in the set, and for every i, $1 \leq i \leq n$, the pair $x_i, y_i \in \mathbb{N}_0$, $0 \leq x_i, y_i \leq 1000$, defines the x- and the y-coordinate of the i-th point. All the points are pairwise different.

Output For every test instance we output only one line. This line contains the list of the coordinates of the vertices of the convex hull in clockwise order, starting from the leftmost point (smallest x coordinate). If two points in the list have the same x-coordinate, then we start from the one with the smallest y-coordinate.

Example

Input:

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

Output:

```
1 1 3 9 2 4
0 0 0 3 2 3 2 0
```

Directions There is only one testset for 100 points in this exercise.