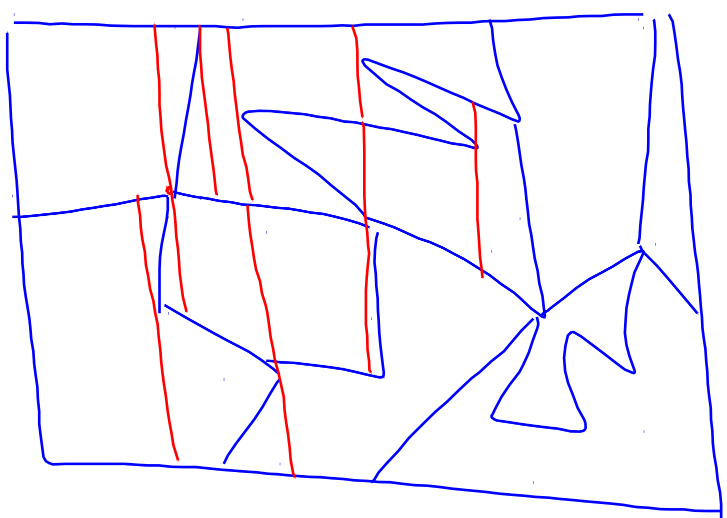
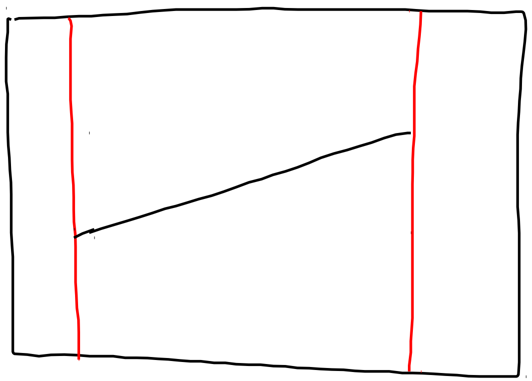
Point location

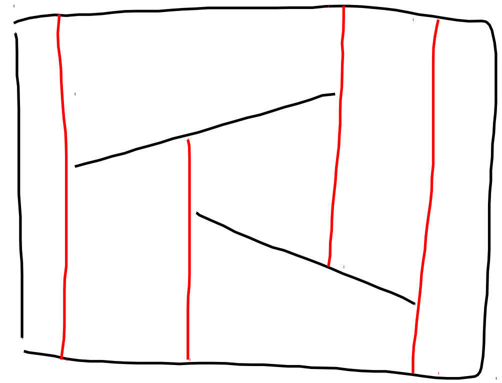
We have a map, we want to construct a searching structure (for this map) which enables us to find the location of any point given by coordinates.



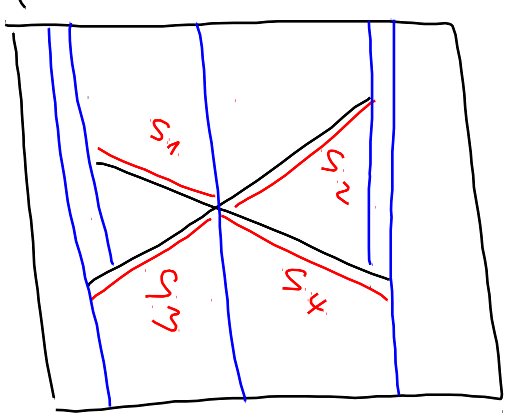
$p = (x_p, y_p)$

Method is to subdivide the given into so called trapezoidal map.





Important observation: We can construct a trapezoidal map for any set of segments (which do not have intersections in inner point. The segments can have common end points.



We suppose that different end points have different x-coordinates!

The description of trapezoids

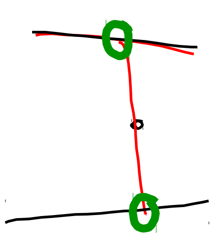
by 4 data. Trapezoid Δ

- top (Δ) segment which forms upper side
- bottom (Δ) —"— —"— lower side
- leftp (Δ) .... end point of a segment from the left
- rightp (Δ) ... end point —"— from the right

top (Δ) and bottom (Δ) determines the face of the original doubly connected edge list.

Lemma Let T be a trapezoidal map for the set S of n segments. Then T has at most $6n+4$ points and $3n+1$ trapezoids.

Points rectangle R has 4 points



n segment has at most $2n$ end points

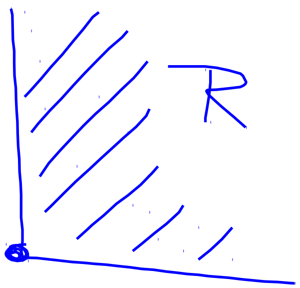
Vertical segment forms new 2 points.

There are at most $2n$ vertical segment

$4+2n+2\cdot 2n$
$=6n+4$

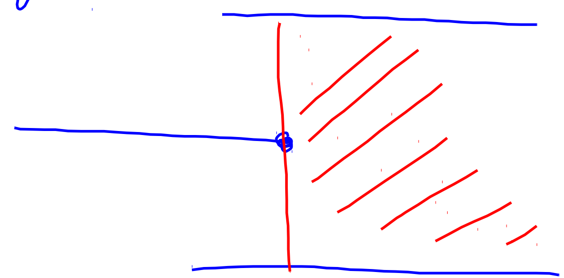
Number of trapezoids

- we count number of trapezoids which have given point as left point



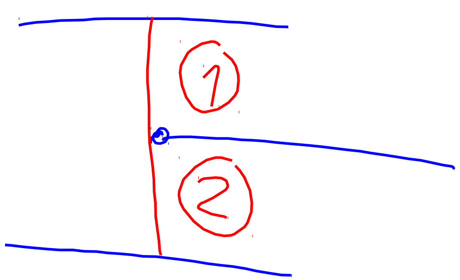
1 trapezoid

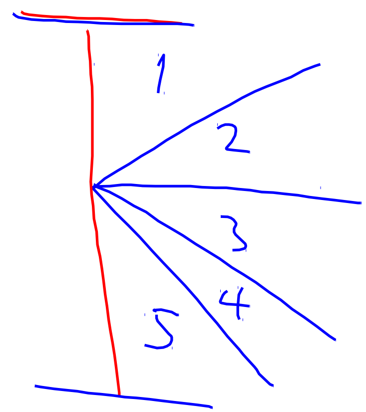
right end point of a segment



determines at most 1 trapezoid

left end point of a segment





end point of 4 segments

Number of trapezoids is

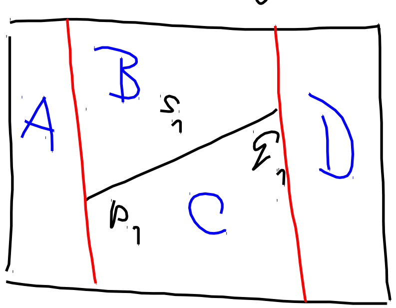
$$\le 1 + n \cdot 1 + n \cdot 2 = 3n+1$$

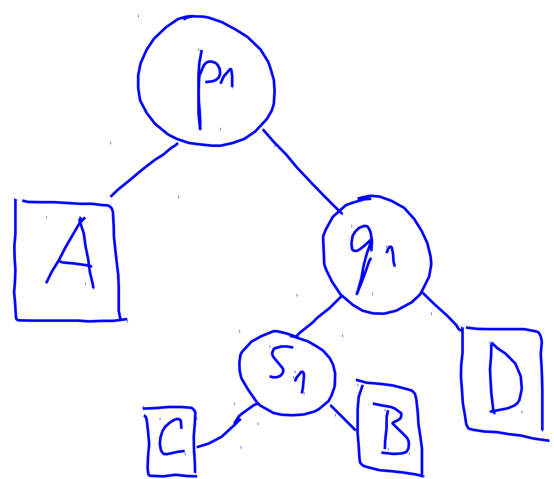
Searching structure

- is an oriented graph
- there are 2 kinds of nodes – end points
  – segments
- from every node different from a leaf there are 2 arrows
- leaves are trapezoids

Searching a point

- if node is an end point then go left if given point lies to the left from this end point
- if node is a segment go left if the point lies under the segment





Picture 8.6 Searching structure for 2 segments.

There is also a mistake in the Czech text. Missing trapezoid F !

Randomized incremental algorithm

$$S_i = \{ s_1, s_2, \dots s_i \}$$

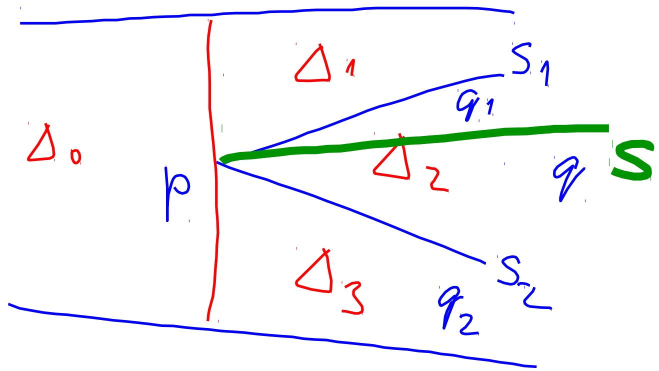
the order of segment is randomized.

We construct a trapezoidal map T_i and corresponding searching structure from trapezoidal map T_{i-1} and a searching structure $\mathcal{D}_{i-1}$.

Algorithm is given by 3 steps.

① Follow segment — finds all trapezoids which are in T_{i-1} intersected by the segment s_i (in order from the left to the right)

Pick 8.8



We find the trapezoid through which s passes using the slopes of segment

$$\frac{q_{1y} - p_y}{q_{1x} - p_x} \geq \frac{q_y - p_y}{q_x - p_x} \geq \frac{q_{2y} - p_y}{q_{2x} - p_x}$$

$\Longrightarrow$ s lies in Δ_2

Left neighbours

2nd step: Replacing intersecting trapezoids by new trapezoids

See pictures 8. 13, 14, 15, 16.

Removing assumptions that the different points have different x- coordinates.

Shear transformation

$$\varphi(x, y) = (x + \varepsilon y, y)$$

We have $p = (x, y_1)$ $\quad p = q \implies y_1 \neq y_2$ (suppose $y_1 < y_2$)

$q = (x, y_2)$

Now $x + \varepsilon y_1 < x + \varepsilon y_2$ and x-coordinate of p and q are different after shear transformation

On the other hand. If p and q have different x-coordinate, then we can find $\varepsilon > 0$ sufficiently small such that p and q have different x-coordinate also after shear transformation.

<u>Conclusion</u>: If we have a finite set of points we can find $\varepsilon > 0$ sufficiently small such that the points have different x-coordinates after shear transformation.

10

Lemma (in text 8.3)

For every finite set P of points and $\varepsilon > 0$ sufficiently small the ordering of points according to $\varphi(p)_x$ is the same as lexicographic ordering of points first with respect to x-coordinate and then with respect to y-coordinate.

$p < q$ in lexicographic ordering

either $p_x < q_x$ $\Rightarrow p_x + \varepsilon p_y < q_x + \varepsilon q_y$

or $p_x = q_x$ and $p_y < q_y$ $\Rightarrow p_x + \varepsilon p_y < q_x + \varepsilon q_y$

Theorem (A) The *expected* running time for searching is $O(\log n)$ for the set of n segments.

(B) The *expected size* of the search structure is $O(n)$.

(C) The *expected* time for the construction of the search structure is $O(n \log n)$.