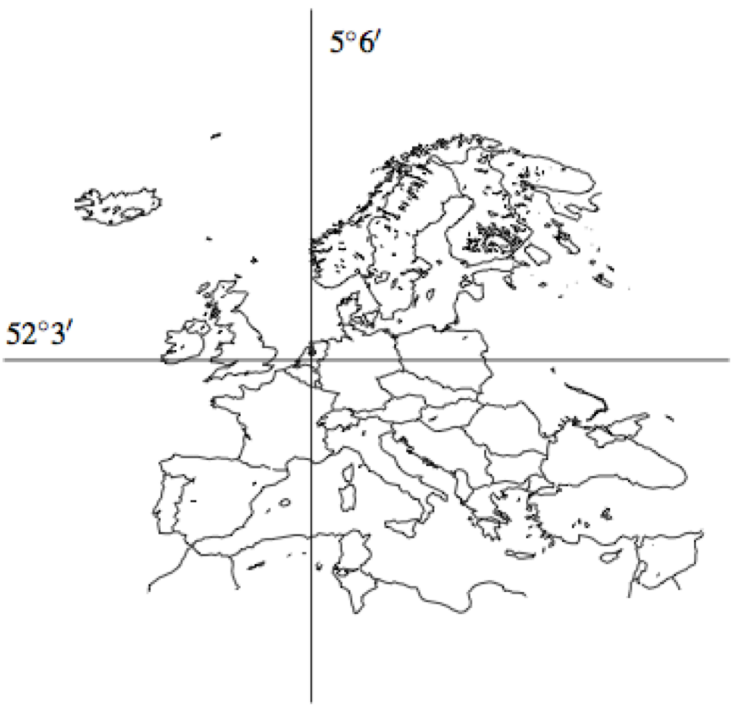
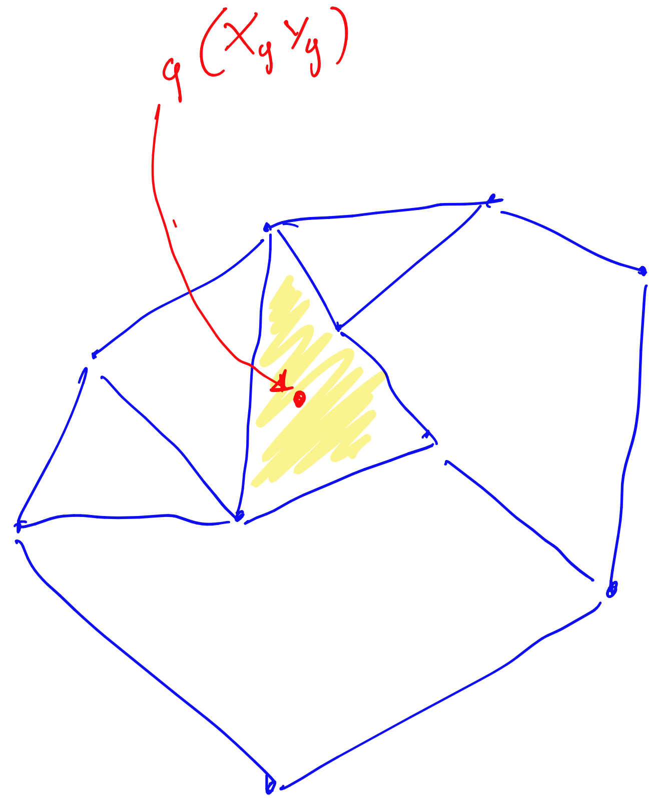
Where are you ?



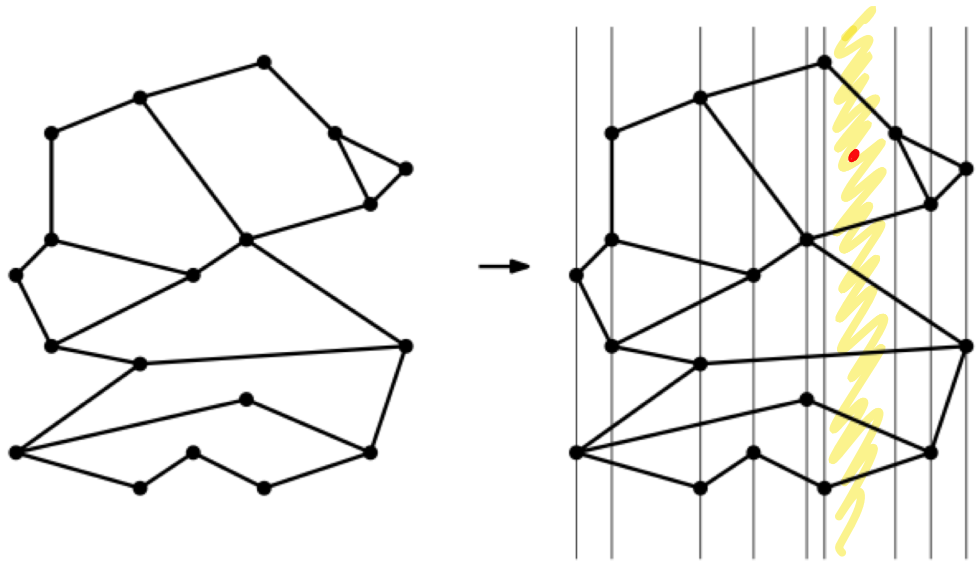


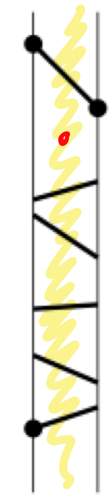
Let S be a planar subdivision with n edges.

The planar point location problem is to store S in such a way that for a given point q, we can report the face f of S that contains q.

Computational Geometry
6 : Point Location
pages 121-144

A first very simple data structure





This procedure divides the plane into vertical slabs

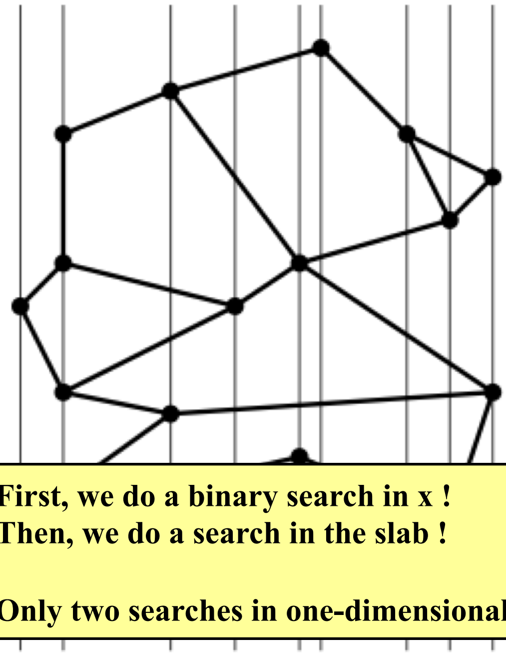
Each slab is then divided by some line segments

We draw vertical lines through all vertices of the subdivision.

This partitions the plane into vertical slabs.

Query time is good !

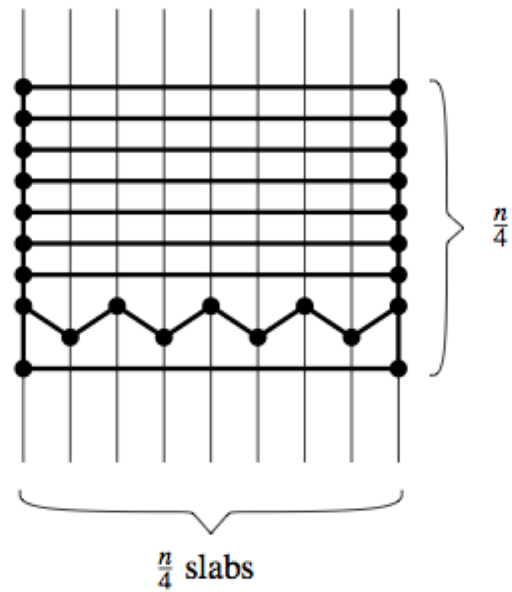
$O(\log n)$



First, we do a binary search in x !
Then, we do a search in the slab !

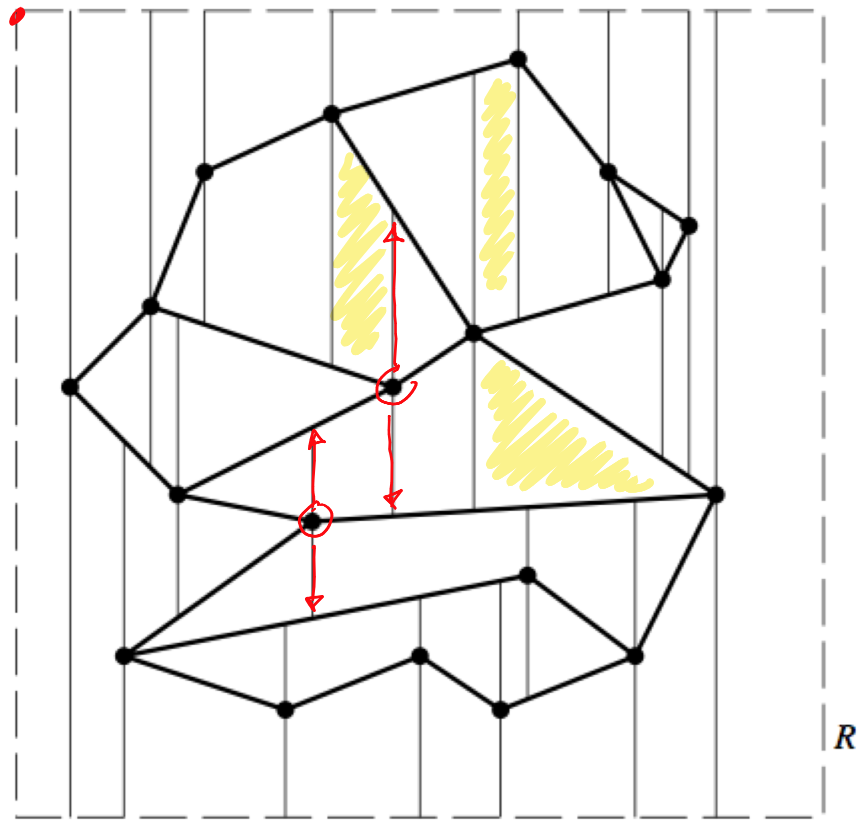
Only two searches in one-dimensional lists

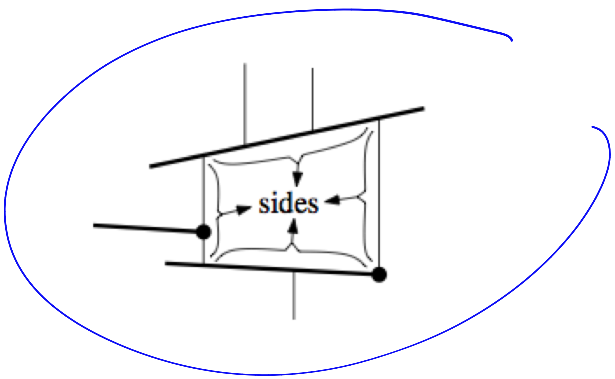
$O(n^2)$



Storage is bad !

Trapezoidal Maps

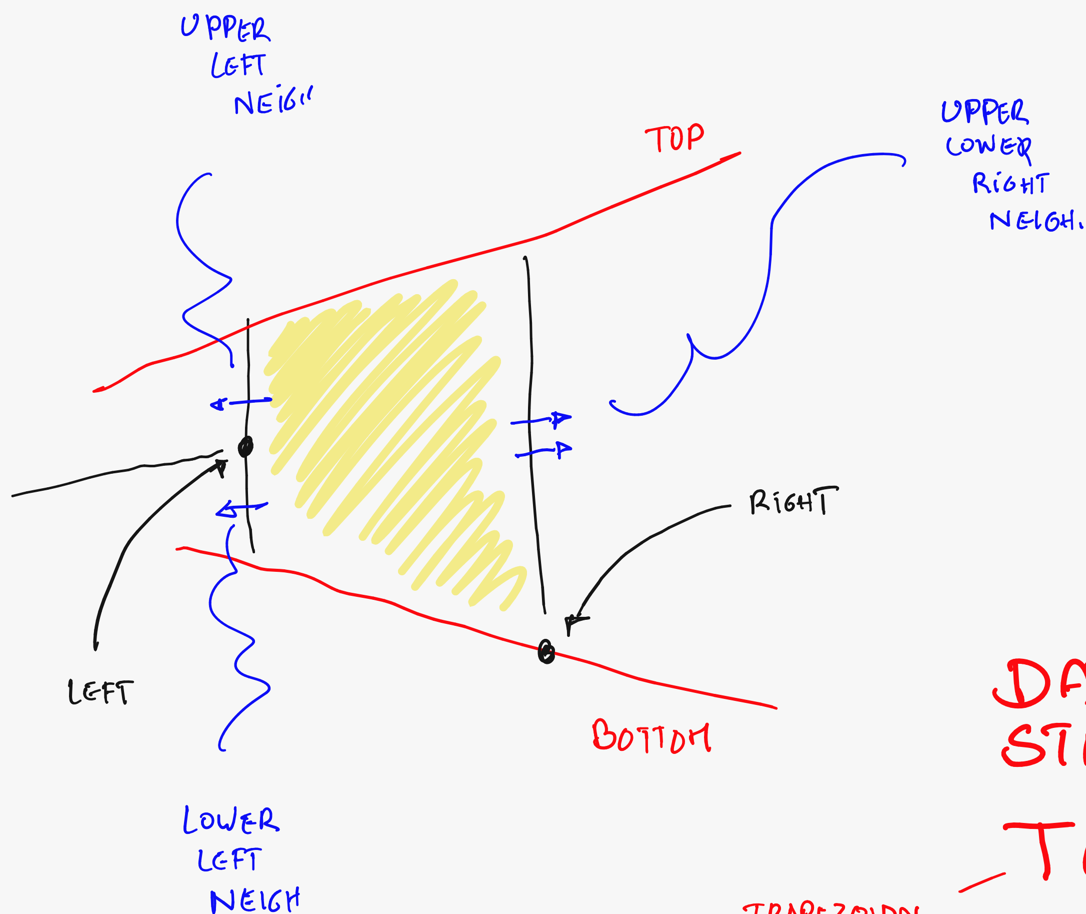




We draw two vertical extensions from every endpoint p of a segment in S, one extension going upwards and one going downwards.

The extensions stop when they meet another segment.

Let S be a set of segments in general position
T(S) is the trapezoidal map of S



DATA STRUCTURE

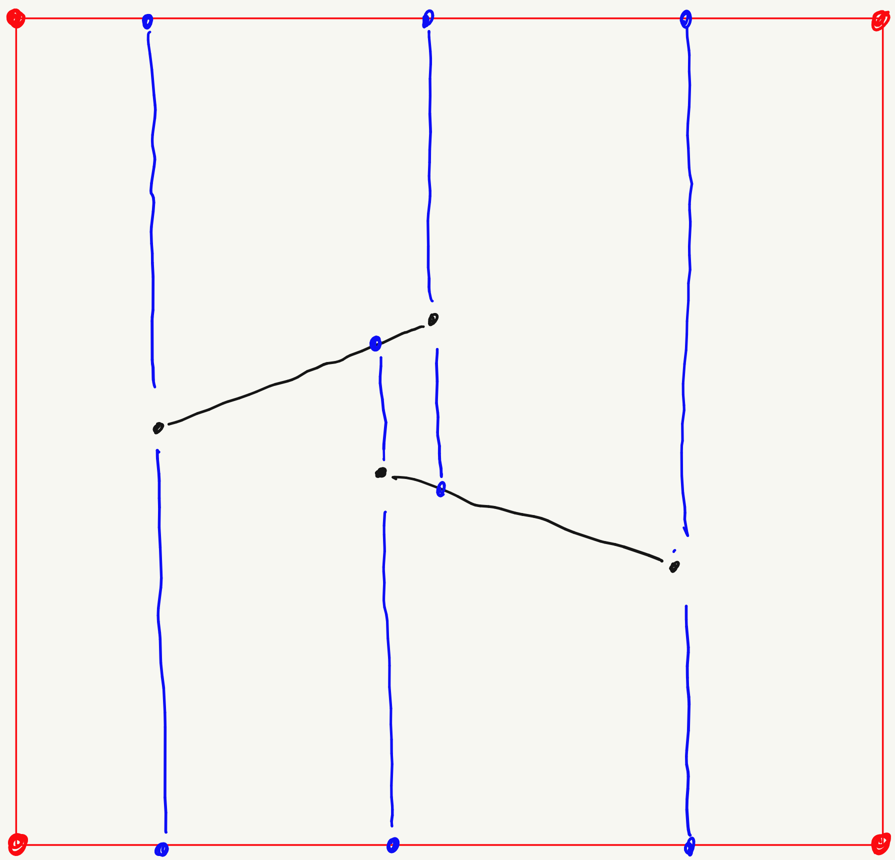
$T(S)$

TRAPEZOIDAL MAP

SEGMENT

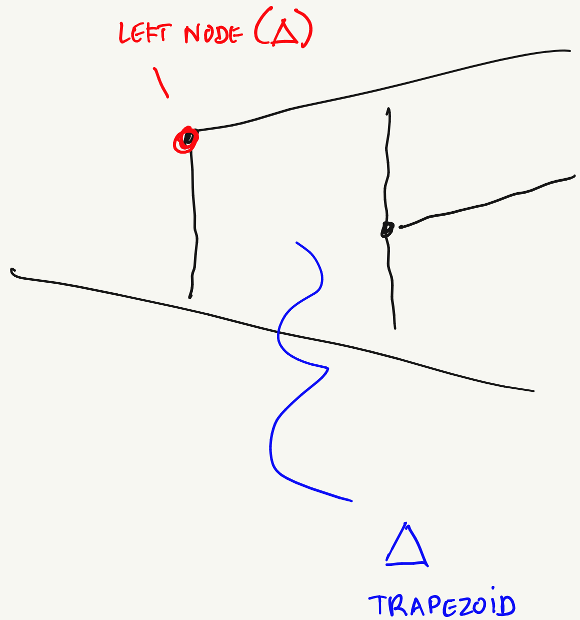
$S = \{ s_0, s_1, s_2, s_3 \}$

SET OF SEGMENTS



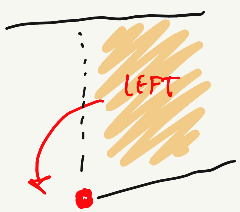
m SEGMENTS

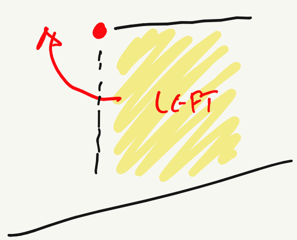
$$\underbrace{4 + 2m + 4m}_{6m+4 \text{ VERTICES}}$$

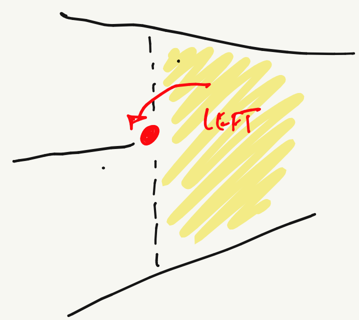


$3n + 1$

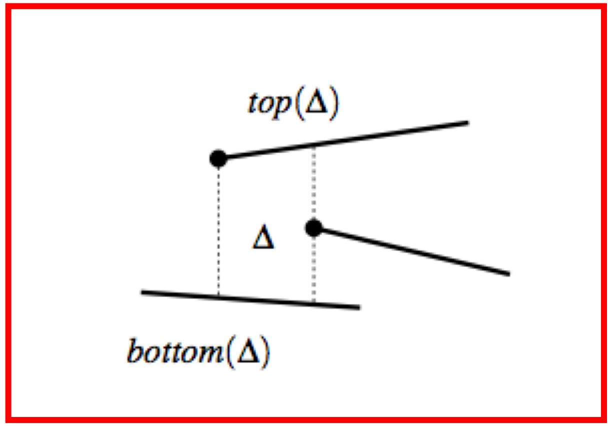
TRAPEZOIDS



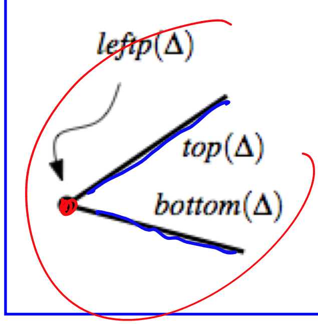




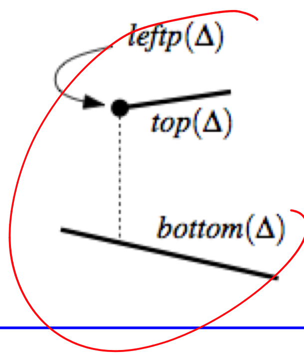
Let us inspect the left side those trapezoids



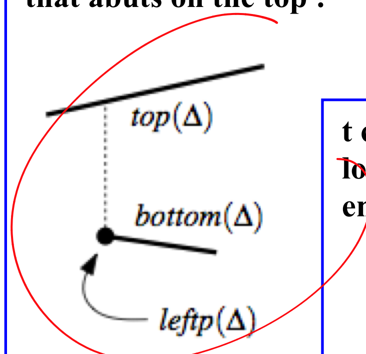
It generates to a point !



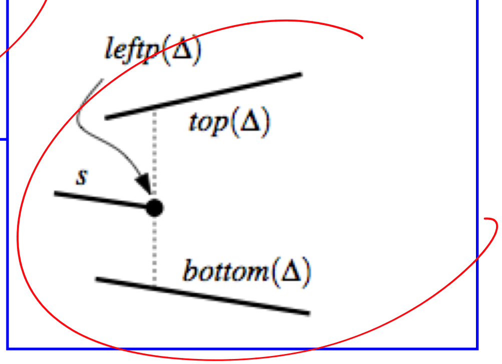
It is the lower vertical extension of the left top that abuts on the bottom !



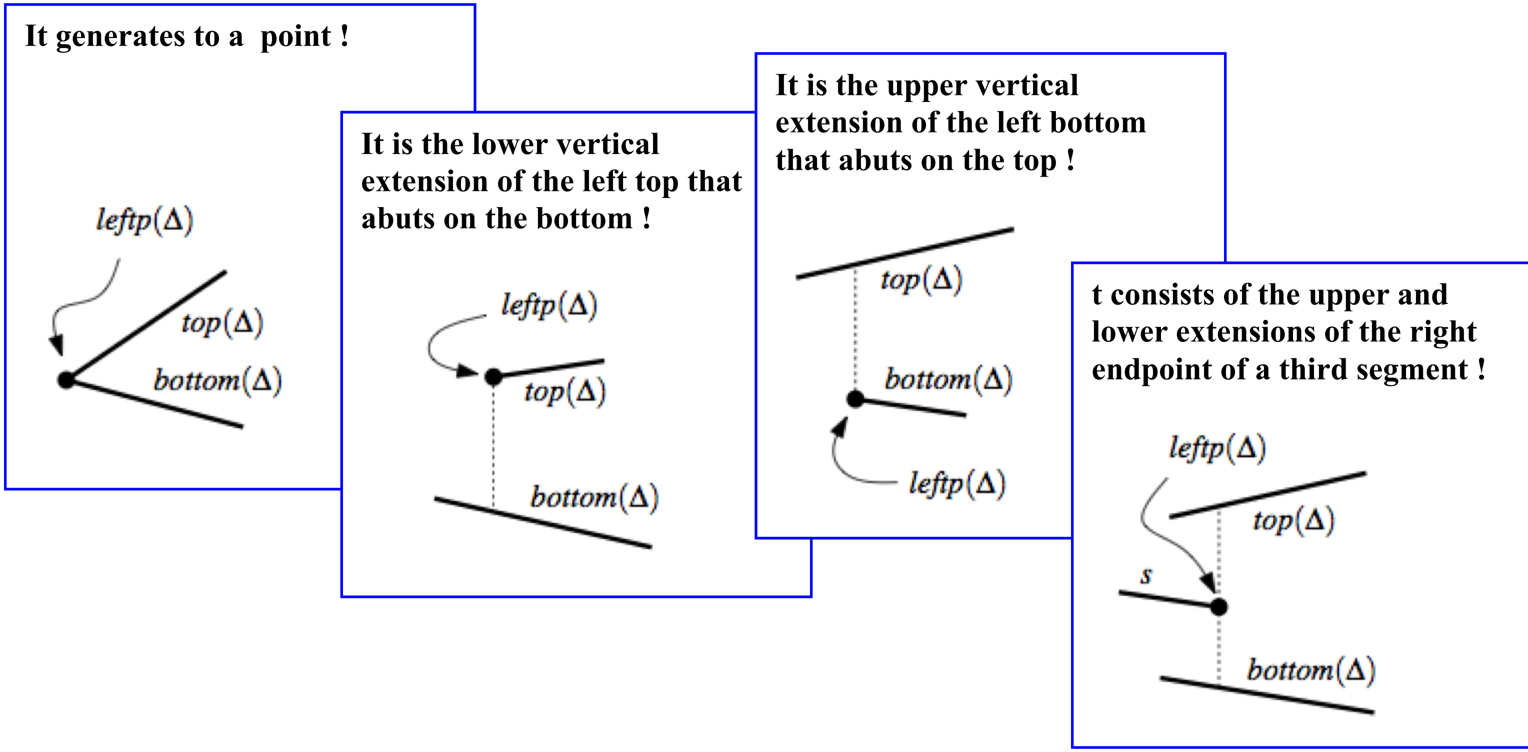
It is the upper vertical extension of the left bottom that abuts on the top !



t consists of the upper and lower extensions of the right endpoint of a third segment !

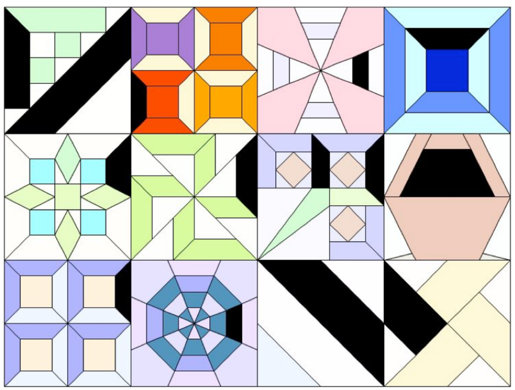


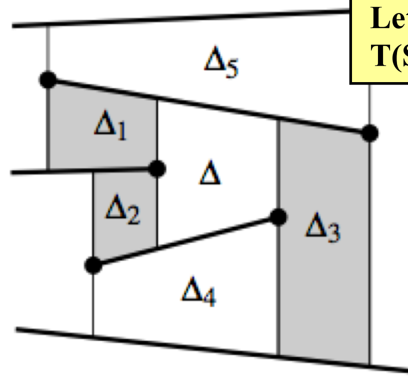
$3n+1$
TRAPEZOIDS



Complexity of the trapezoidal maps

Lemma 6.2 *The trapezoidal map $\mathcal{T}(S)$ of a set S of n line segments in general position contains at most $6n+4$ vertices and at most $3n+1$ trapezoids.*



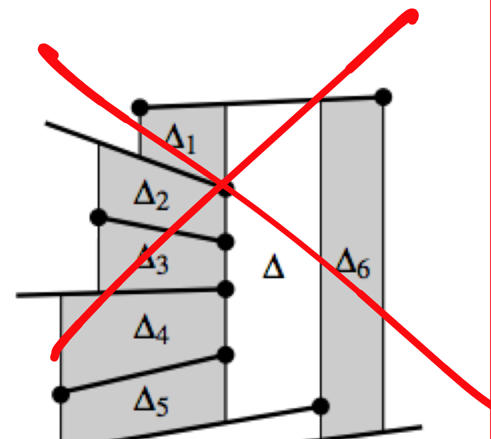


Let S be a set of segments in general position
T(S) is the trapezoidal map of S

OK

NODES IN GENERAL POSITION

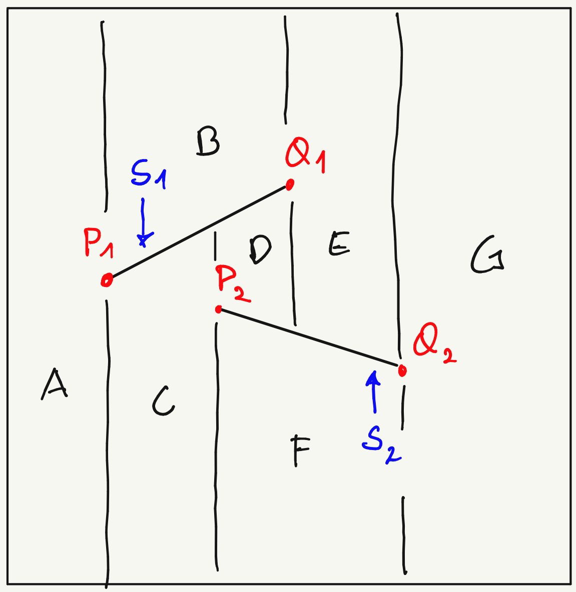
Adjacents Trapezoids

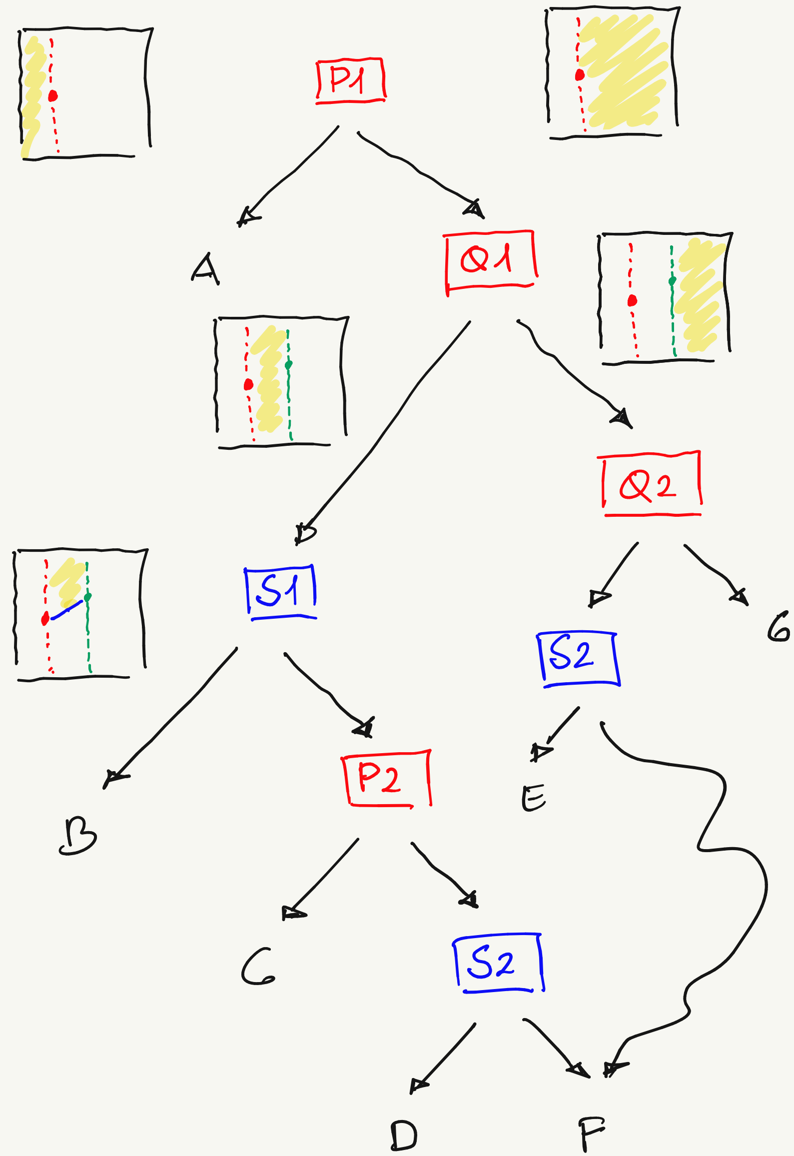


Let S be a set of segments not in general position
T(S) is the trapezoidal map of S

Specialized data structure for each trapezoid

- **Top, Bottom, Leftp, Rightp**
- **Pointers to its at most four neighbors**


B
Q_1
S_1
P_1
D
E
G
P_2
Q_2
A
C
F
S_2


P1
A
Q1
S1
Q2
S2
G
B
P2
E
C
S2
D
F

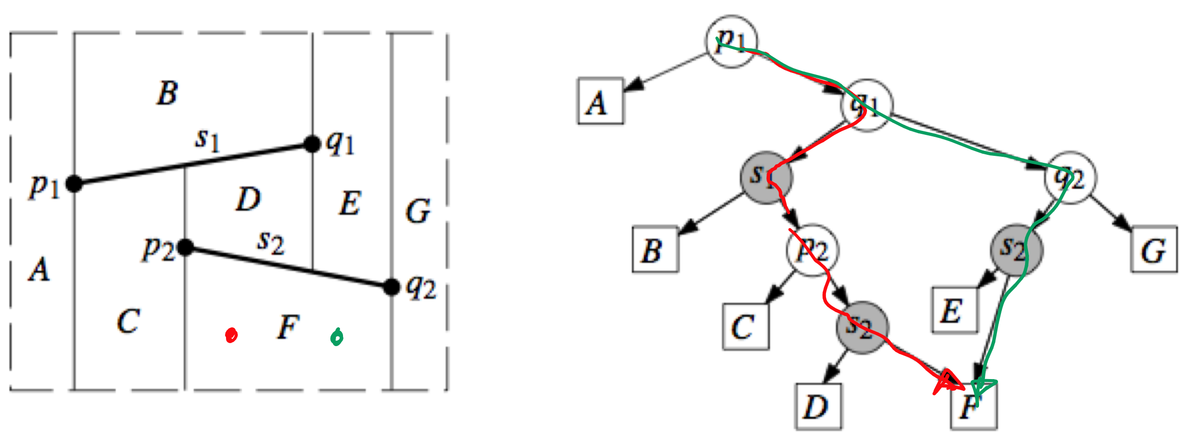
A directed acyclic graph as the search structure

D(S)

DATA SEARCH STRUCTURE

The search structure D and the trapezoidal map T(S) computed by the algorithm are interlinked.

- **A trapezoid of T(S) has a pointer to the corresponding leaf of D.**
- **A leaf node of D has a pointer to the corresponding trapezoid in T(S).**



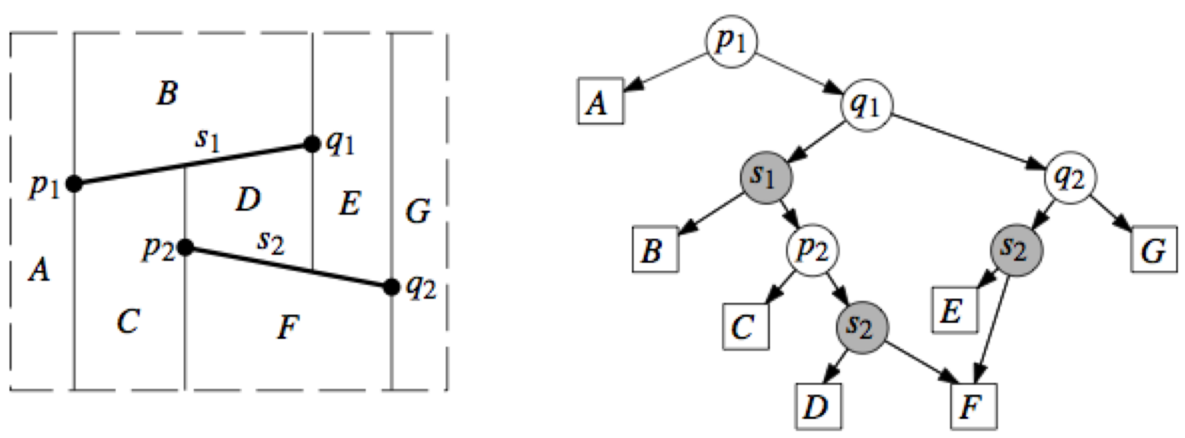
A directed acyclic graph as the search structure

White nodes : nodes

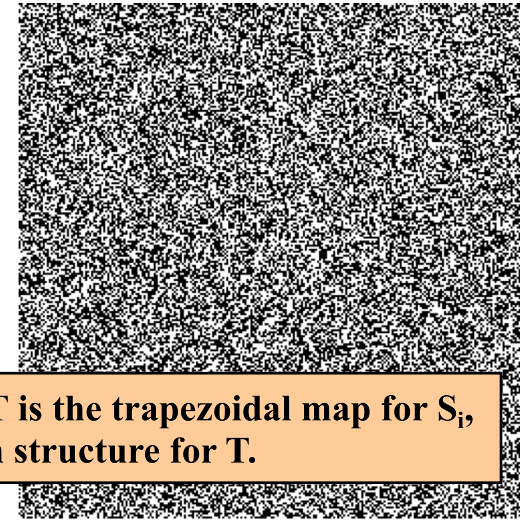
Does q lie to the left or to the right of the vertical line through the endpoint stored at this node ?

Gray nodes : segments

Does q lie above or below the segment s stored here ?



A Randomized Incremental Algorithm



The loop invariant is that T is the trapezoidal map for S_i, and that D is a valid search structure for T.

Algorithm TRAPEZOIDALMAP(S)

Input. A set S of n non-crossing line segments.

Output. The trapezoidal map $\mathcal{T}(S)$ and a search structure $\mathcal{D}$ for $\mathcal{T}(S)$ in a bounding box.

1. Determine a bounding box R that contains all segments of S, and initialize the trapezoidal map structure $\mathcal{T}$ and search structure $\mathcal{D}$ for it.
2. Compute a random permutation $s_1, s_2, \ldots, s_n$ of the elements of S.
3. **for** $i \leftarrow 1$ **to** n
4.   **do** Find the set $\Delta_0, \Delta_1, \ldots, \Delta_k$ of trapezoids in $\mathcal{T}$ properly intersected by s_i.
5.   Remove $\Delta_0, \Delta_1, \ldots, \Delta_k$ from $\mathcal{T}$ and replace them by the new trapezoids that appear because of the insertion of s_i.
6.   Remove the leaves for $\Delta_0, \Delta_1, \ldots, \Delta_k$ from $\mathcal{D}$, and create leaves for the new trapezoids. Link the new leaves to the existing inner nodes by adding some new inner nodes, as explained below.

Algorithm TRAPEZOIDALMAP(S)

Input. A set S of n non-crossing line segments.

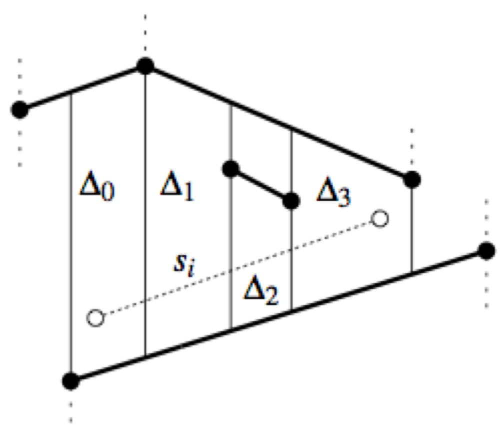
Output. The trapezoidal map $\mathcal{T}(S)$ and a search structure $\mathcal{D}$ for $\mathcal{T}(S)$ in a bounding box.

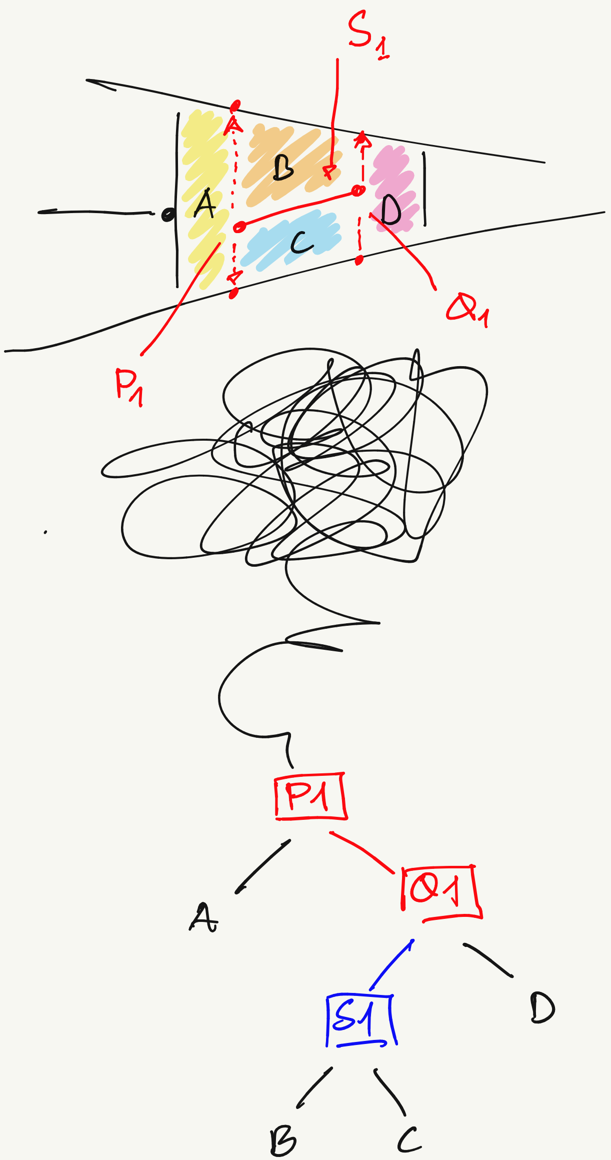
1. Determine a bounding box R that contains all segments of S, and initialize the trapezoidal map structure $\mathcal{T}$ and search structure $\mathcal{D}$ for it.
2. Compute a random permutation $s_1, s_2, \ldots, s_n$ of the elements of S.
3. **for** $i \leftarrow 1$ **to** n
4.   **do** Find the set $\Delta_0, \Delta_1, \ldots, \Delta_k$ of trapezoids in $\mathcal{T}$ properly intersected by s_i.
5.   Remove $\Delta_0, \Delta_1, \ldots, \Delta_k$ from $\mathcal{T}$ and replace them by the new trapezoids that appear because of the insertion of s_i.
6.   Remove the leaves for $\Delta_0, \Delta_1, \ldots, \Delta_k$ from $\mathcal{D}$, and create leaves for the new trapezoids. Link the new leaves to the existing inner nodes by adding some new inner nodes, as explained below.

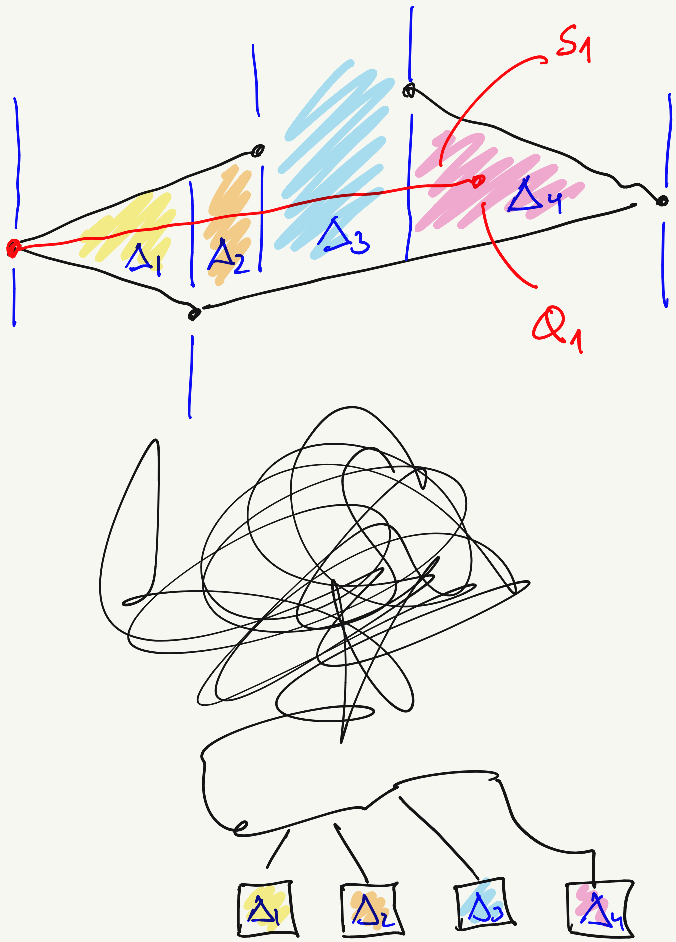
How to insert a segment in lines 4-6 ?

To modify the current trapezoidal map, we first have to know where it changes. This is exactly at the trapezoids that are intersected by the segments.

Our first task is therefore to find the intersected trapezoids.

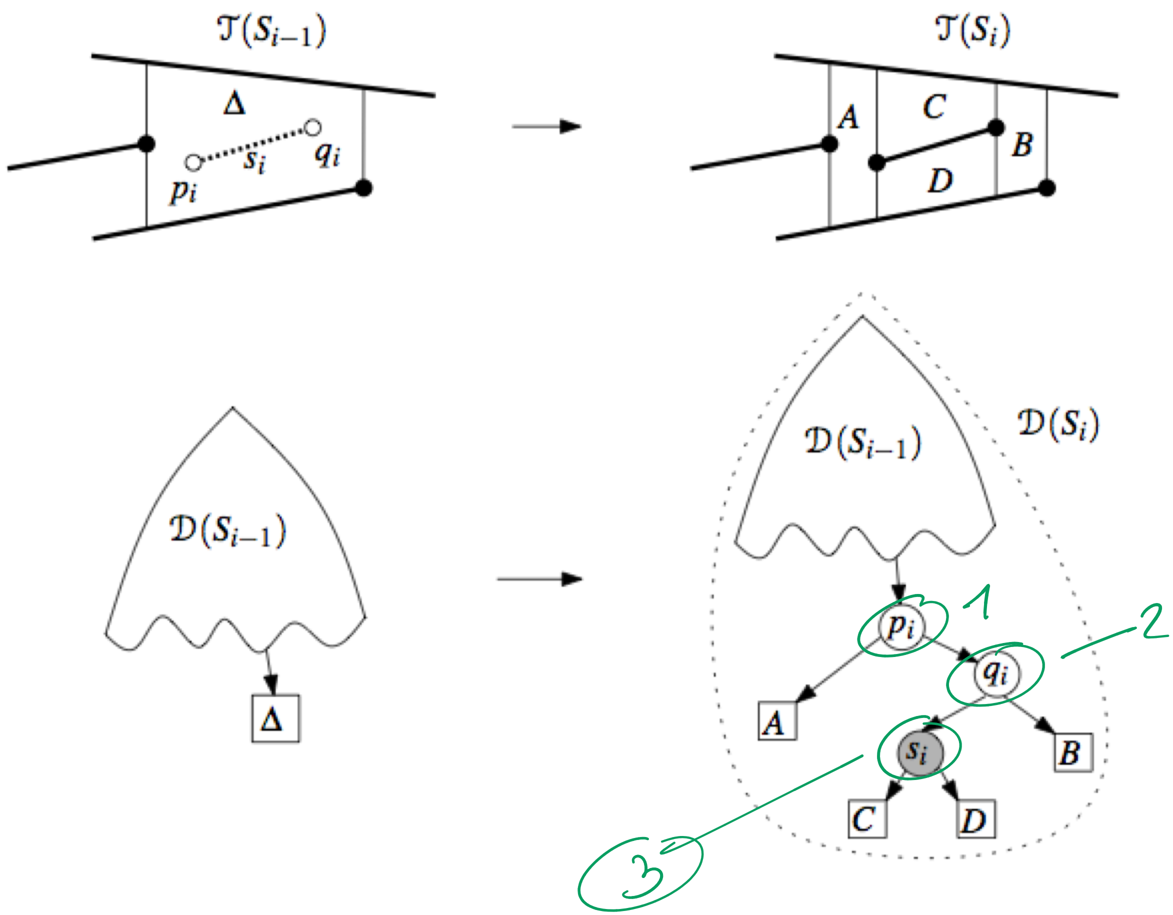


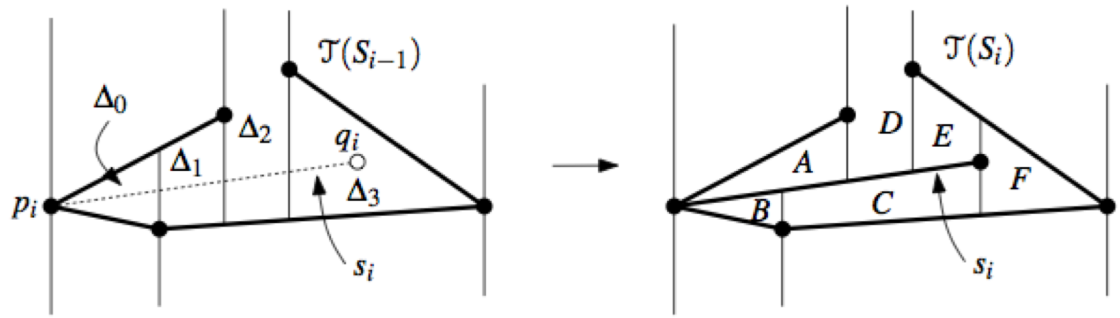

S1
A
B
C
D
P1
Q1
P1
A
Q1
S1
D
B
C

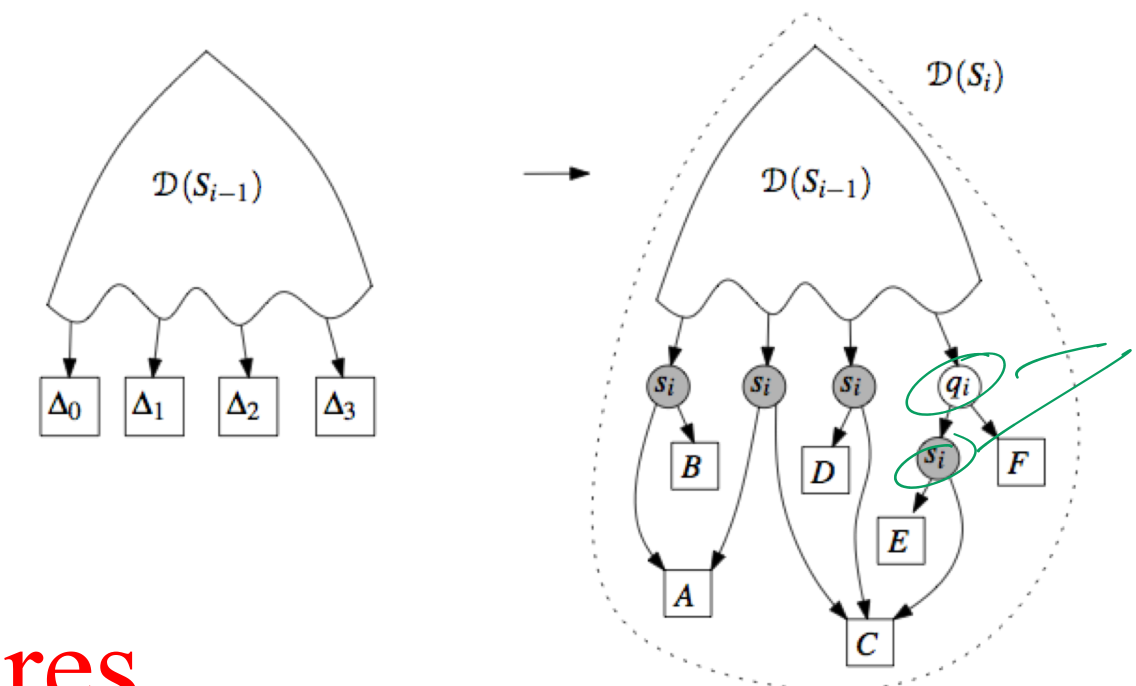

S1
Q1
Δ1
Δ2
Δ3
Δ4


D
E
A
F
B
C
S1
S1
S1
Q1
A
B
D
F
S1
E
C

Updating both data structures







Updating both data structures

QUERY $O(\log n)$ ✓
SIZE $O(n)$ ✓
BUILDING TIME $O(n \log n)$ ✓

SIZE OF D
INCREASES OF 3

$3n$ IS THE BEST POSSIBLE
WORST CASE BOUND !

AVERAGE QUERY TIME $\neq$ AVERAGE MAX QUERY TIME

EXPECTED PERFORMANCES

X_i

NUMBER
OF NODES
ON THE PATH CREATED
AT ITERATION i

$$\underbrace{E\left(\sum_i X_i\right)}_{\text{EXPECTED PATH LENGTH}} = \sum_i \underbrace{E(X_i)}_{\leq 3 P_i}$$

ITERATION i
CONTRIBUTES TO
THE PATH SEARCH

$$P_i = \text{PROBABILITY}\left(\Delta_q(S_{i-1}) \neq \underbrace{\Delta_q(S_i)}_{\notin T(S_{i-1})}\right)$$

$S_{i-1} = \{s_0 \dots s_{i-1}\}$

$= \frac{4}{i}$

$$\underbrace{E\left(\sum_i X_i\right)}_{\substack{\text{EXPECTED}\\ \text{PATH}\\ \text{LENGTH}}} = \sum_i \underbrace{E(X_i)}_{\leq 3P_i}$$

ITERATION i CONTRIBUTES TO THE PATH SEARCH

$$P_i = \text{PROBABILITY}\left(\Delta_q(S_{i-1}) \neq \underbrace{\Delta_q(S_i)}_{\notin T(S_{i-1})}\right)$$

$$S_{i-1} = \{s_0 \dots s_{i-1}\}$$

$$= \frac{4}{i}$$

$$\underbrace{E\left(\sum_i X_i\right)}_{O(\log n)} = \sum_i 3P_i = \sum_i \frac{12}{i} = 12 \underbrace{\sum_i \frac{1}{i}}_{\log n \leq \cdot \leq \log n + 1}$$

$$\text{SIZE}(D) = \boxed{O(n)} + \boxed{\text{NUMBER OF INNER NODES CREATED AT ALL STAGES OF THE ALGORITHM}}$$

LEAVES
% TRAPEZOIDS :-)

$$\sum_{i=1}^{n} O(i)$$

$O(1)$

NEW LEAVES CREATED AT STAGE i

$$= \sum_{i=1}^{n} E(k_i)$$

$$= \underbrace{\frac{1}{i} \sum_{s \in S_i} \sum_{\Delta \in T(S_i)} \underbrace{S(\Delta, s)}_{}}_{O(i)}$$

1 IF Δ DISAPPEARS WHEN s IS REMOVED FROM S_i

Complexity of the trapezoidal map algorithm

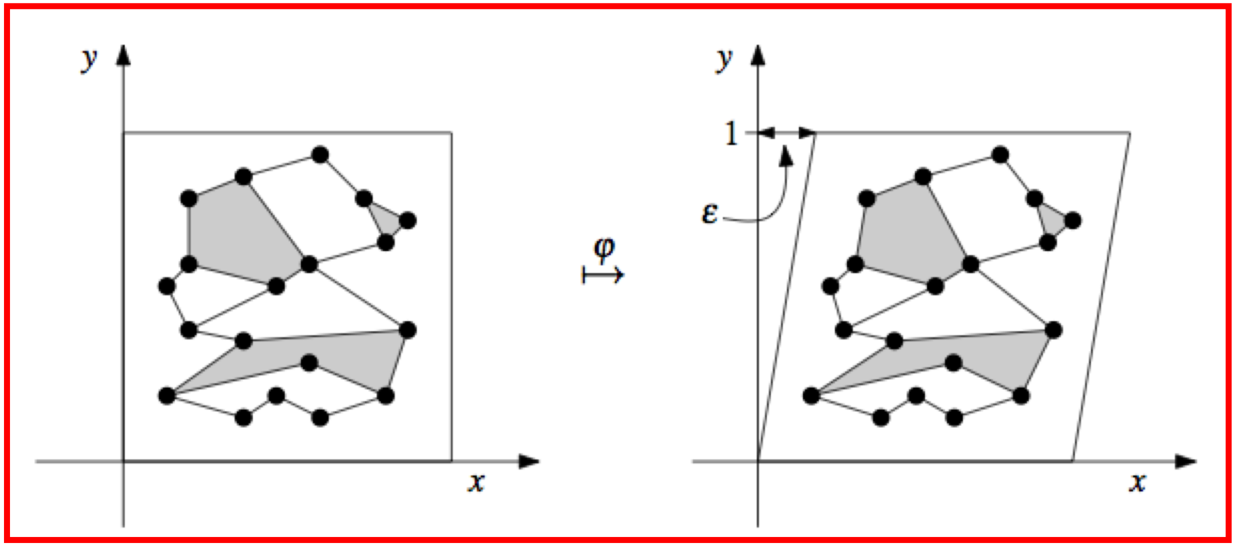
$$\mathrm{E}[k_i] = \frac{1}{i}\sum_{s\in S_i}\sum_{\Delta\in\mathcal{T}(S_i)}\delta(\Delta,s) \leqslant \frac{O(i)}{i} = O(1).$$

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map $\mathcal{T}(S)$ of a set S of n line segments in general position and a search structure $\mathcal{D}$ for $\mathcal{T}(S)$ in $O(n\log n)$ expected time. The expected size of the search structure is $O(n)$ and for any query point q the expected query time is $O(\log n)$.*

Dealing with Degenerate Cases

$$\varphi : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x + \varepsilon y \\ y \end{pmatrix}$$

Composite numbers are used to deal with the case where points have the same coordinate.
Let us have another look at such a symbolic perturbation, and will try to interpret it geometrically.



Composite Number Space

$$\boxed{p := (p_x, p_y)}$$

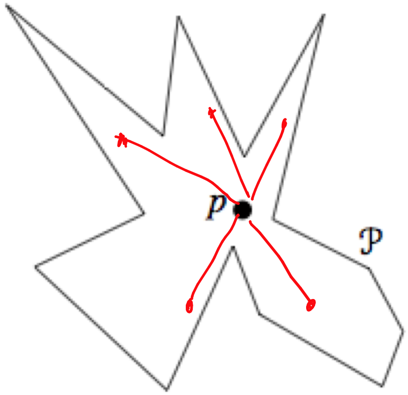
$$\boxed{\hat{p} := ((p_x|p_y),(p_y|p_x))}$$

$$\boxed{[x : x'] \times [y : y']}$$

$$\boxed{[(x|-\infty) : (x'|+\infty)] \times [(y|-\infty) : (y'|+\infty)].}$$

The first coordinate of any two composite points are distinct
The same holds true for the second coordinate.
We construct kd-trees and range trees for this space with the order defined by

$$(a|b) < (a'|b') \quad \Leftrightarrow \quad a < a' \text{ or } (a = a' \text{ and } b < b').$$



Exercice 6

6.7 A polygon $\mathcal{P}$ is called *star-shaped* if a point p in the interior of $\mathcal{P}$ exists such that, for any other point q in $\mathcal{P}$, the line segment $\overline{pq}$ lies in $\mathcal{P}$. Assume that such a point p is given with the star-shaped polygon $\mathcal{P}$. As in the previous two exercises the vertices of $\mathcal{P}$ are given in sorted order along the boundary in an array. Show that, given a query point q, it can be tested in time $O(\log n)$ whether q lies inside $\mathcal{P}$. What if $\mathcal{P}$ is star-shaped, but the point p is not given?