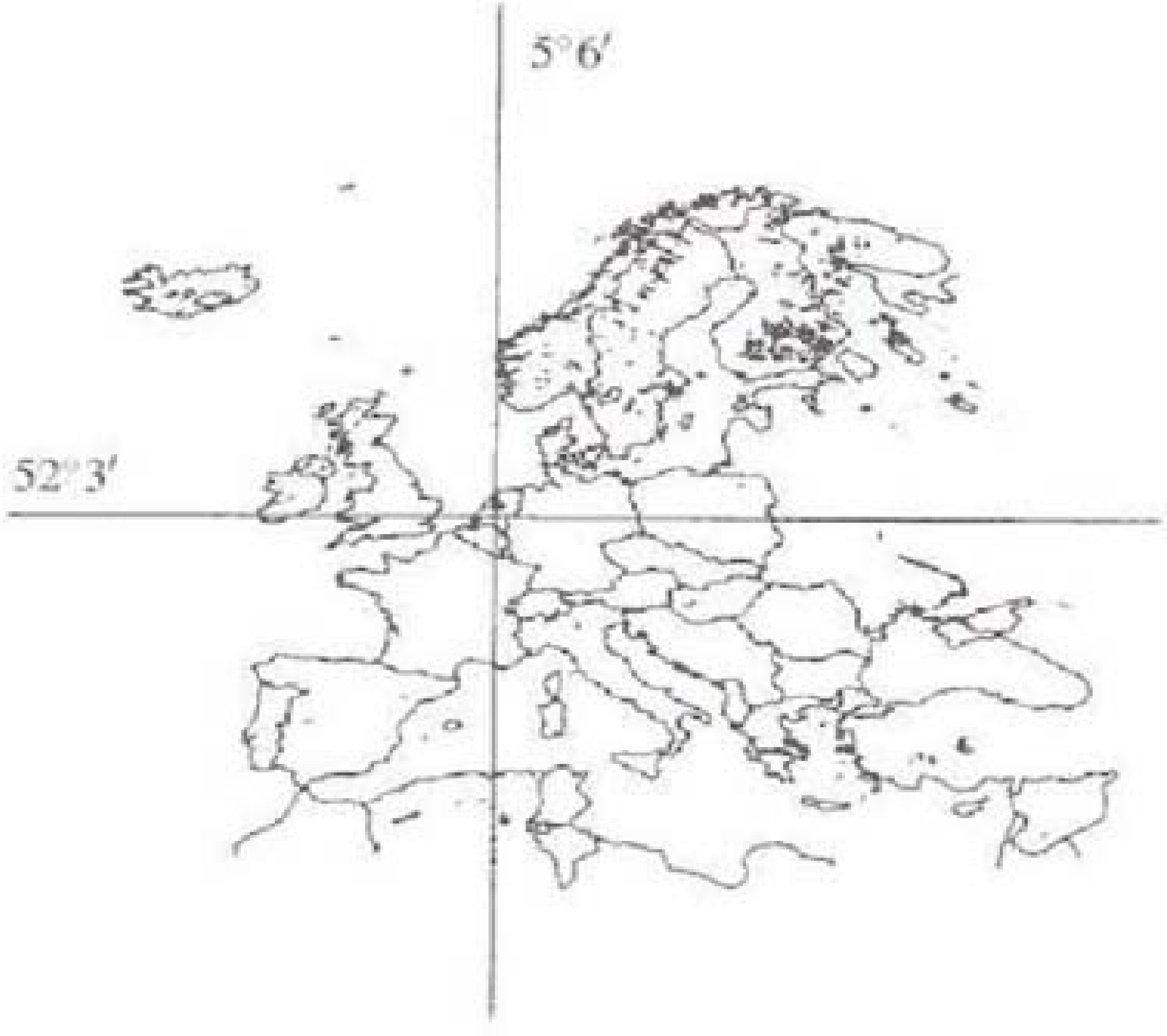
In The Name Of GOD

Point Location

Point Location



Definition

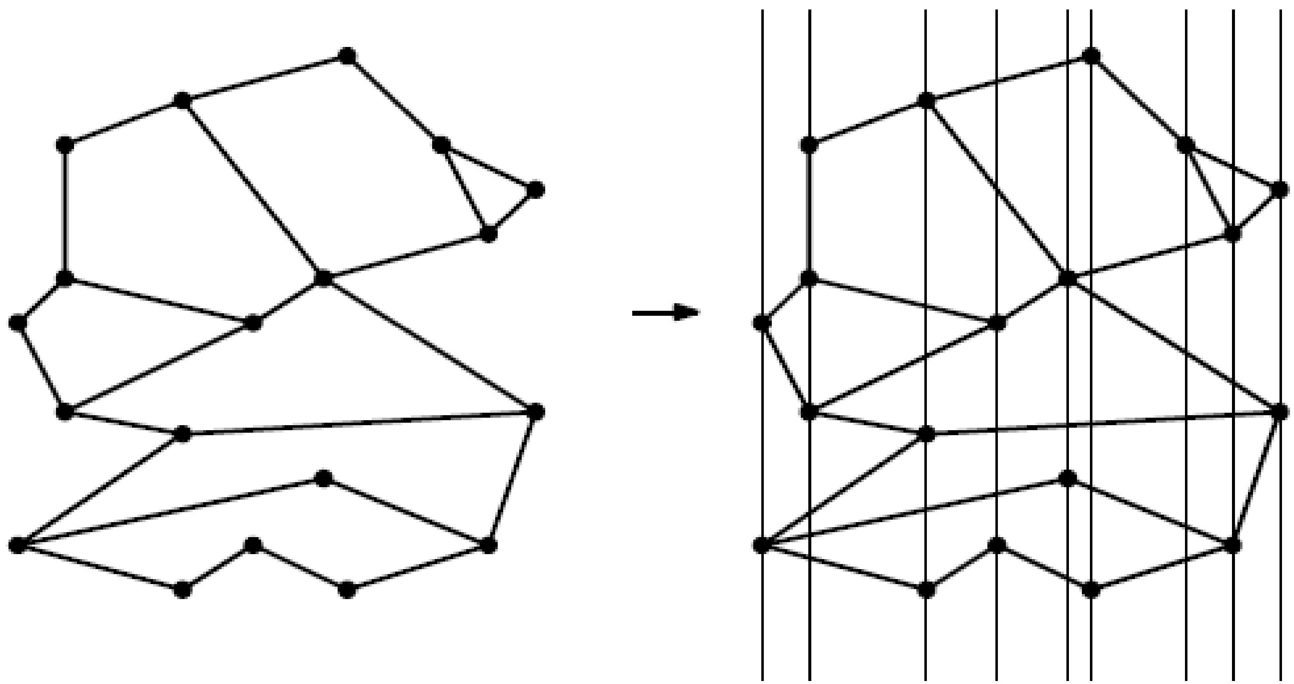
- Given: a planar subdivision S with n edges formed by multiple polygons called faces.

- Goal: build a data structure that, given a query Point, determines which face of the planar Subdivision that point lies in.

- Several different approaches lead to optimal data structures, with O (n) storage space and O (log n) query time.

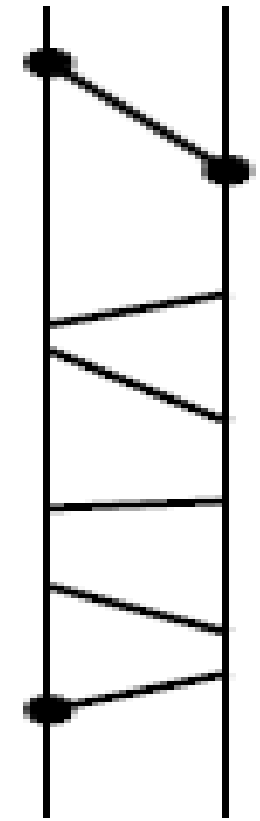
First attempt

- Want to divide the plane into easily manageable sections with simple data structure.
- Idea: Divide the graph into slabs, by drawing a vertical line Through every vertex of the graph.



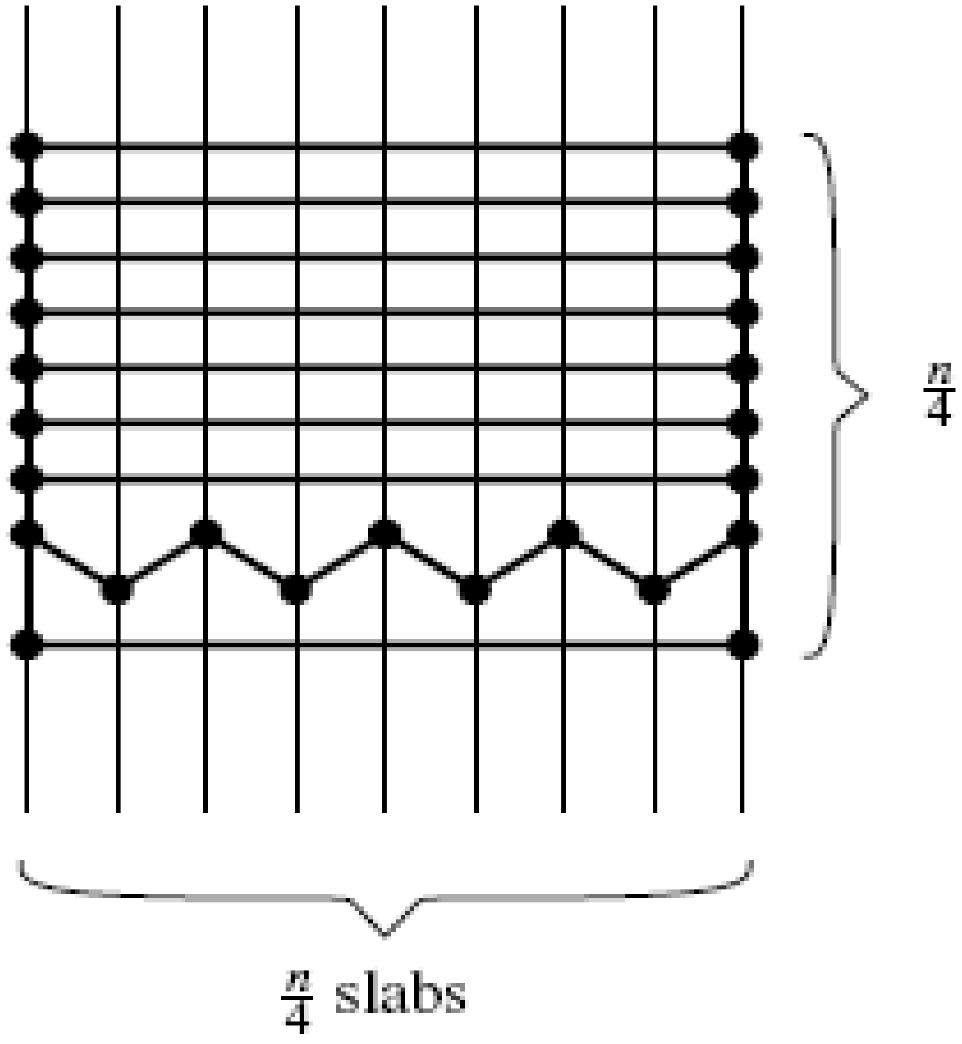
• Given a subdivision of the plane into vertical slabs (area between two parallel); determine which slab contains a given point (do binary search).

• Given the query point, do binary search in the proper slab.



Analysis

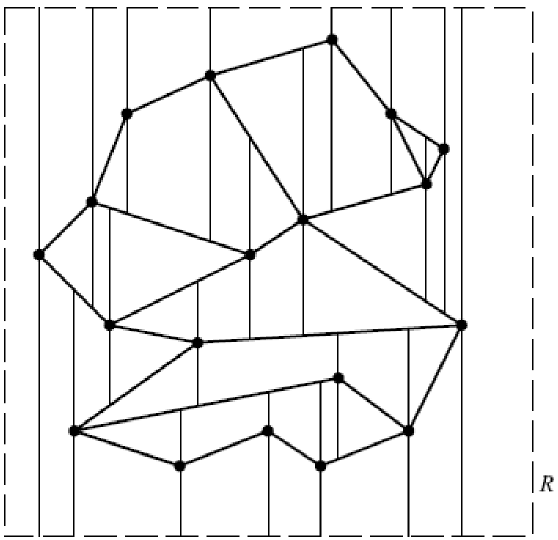
- Query time: $O(\log n)$
- Space: $O(n^2)$

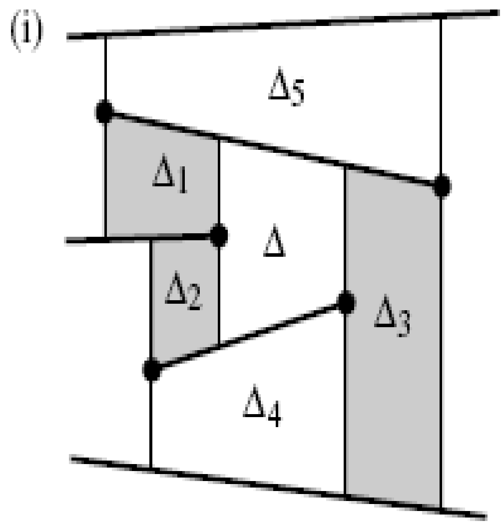


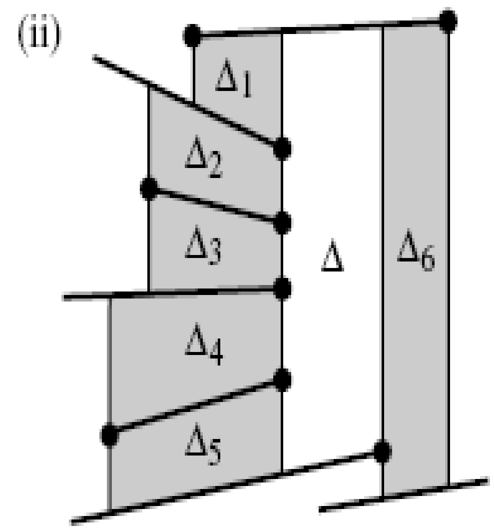
✓Two line segment is non-crossing if their intersection is either empty or comment end point.

Assumptions/Simplifications

- Add a bounding box R that contains S
- Assume that no two distinct endpoints of segments in the set s have the same x-coordinate.



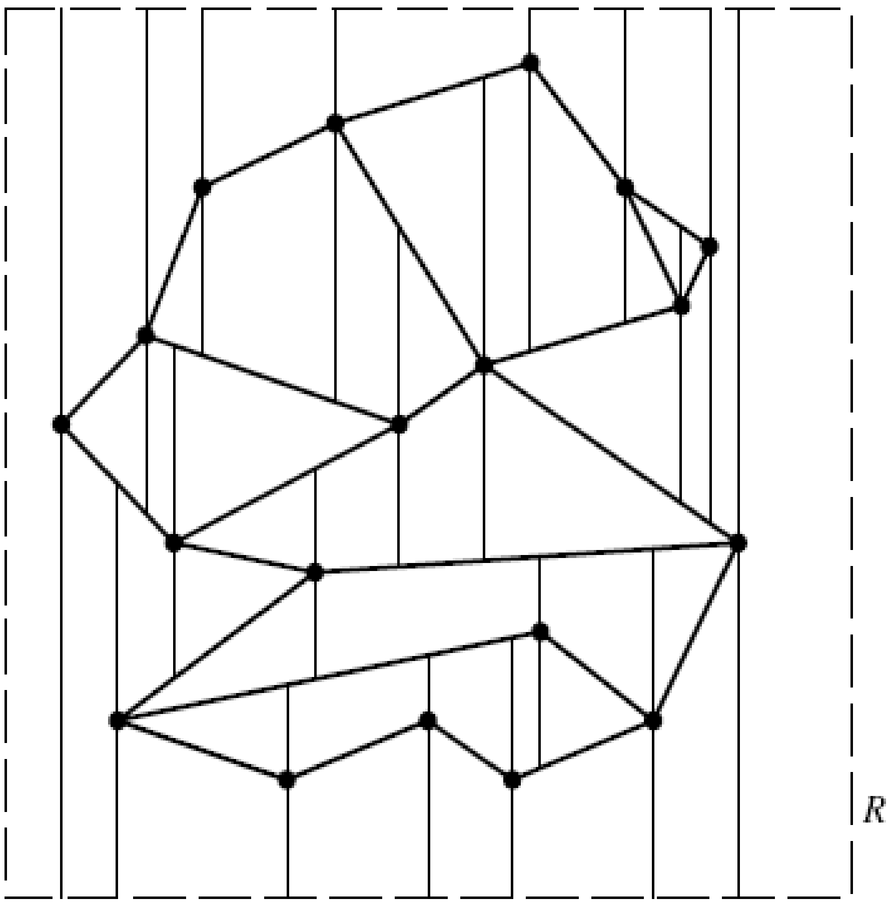




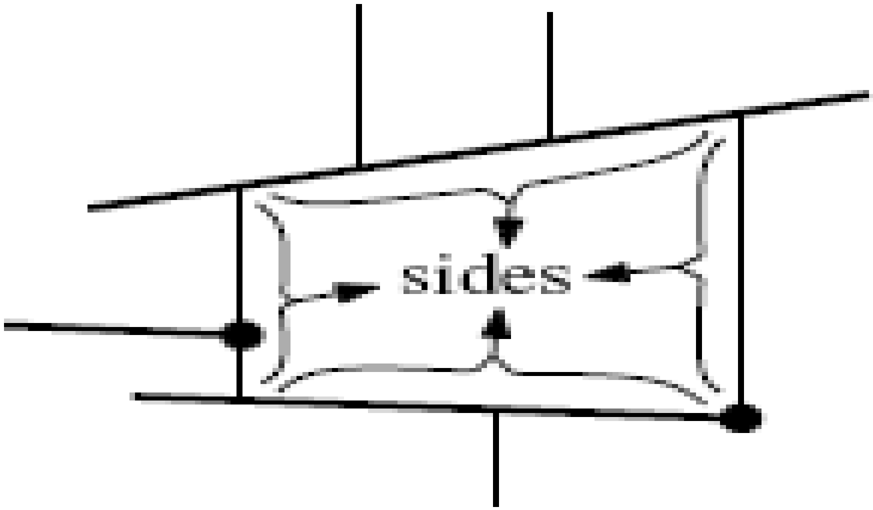
We have a set of n non-crossing line segment ,enclosed in a bounding box R, and with the property that no two distinct endpoint line on a common vertical that call such a set a set of line segment in **general position.**

Second attempt

- We get a trapezoidal decomposition T(S) of S
- A trapezoidal decomposition is obtained by shooting vertical bullets going both up and down from each vertex in the original subdivision. The bullets stop when they hit an edge, and form a new edge in the subdivision.



Lemma 6.1 each face in a trapezoidal map of a set S of line segment in general position has **one or two vertical slides** and **exactly two non-vertical slides.**

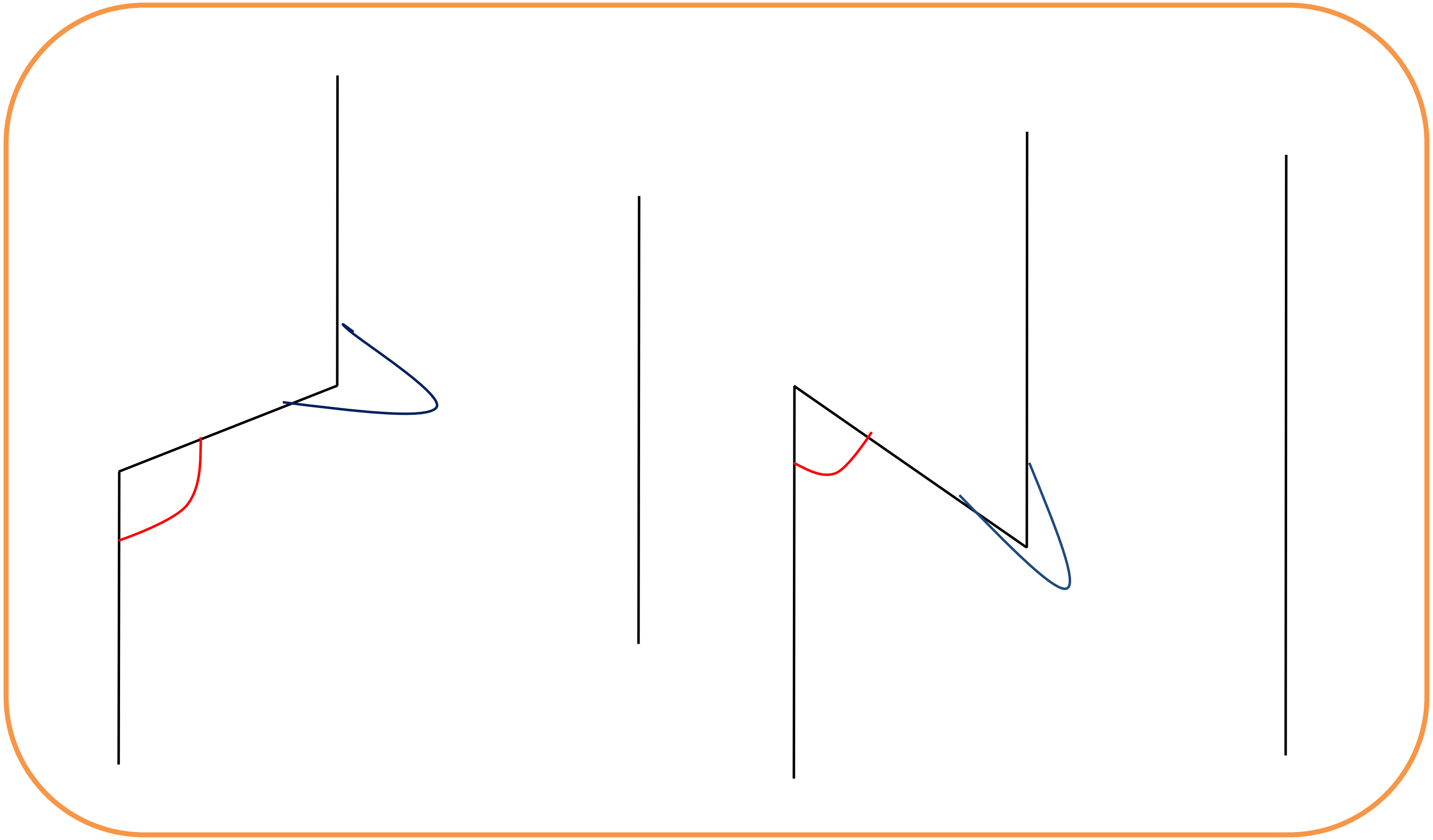


Convex Polygon

- *Definition: A polygon that has all interior angles less than 180°.*
- In geometry, The following property of a simple polygon are all equivalent to convexity:

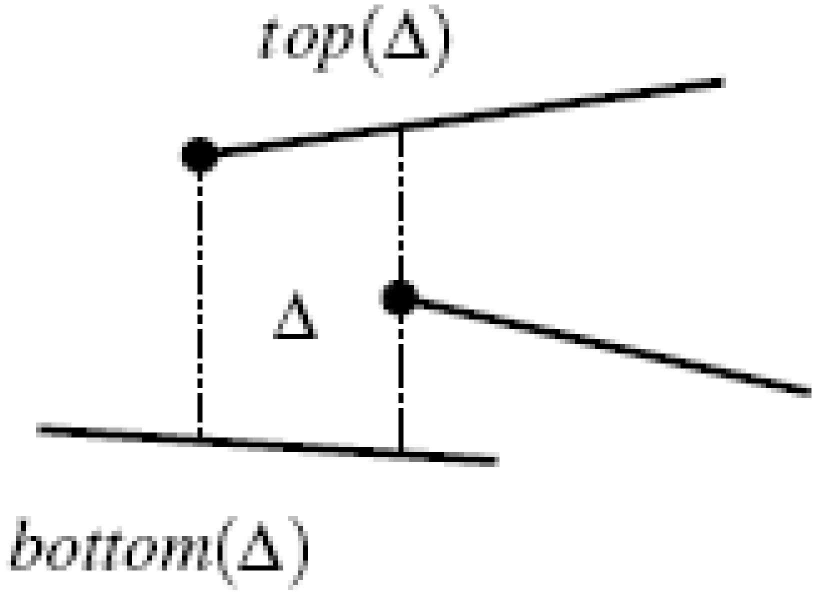
✓Every internal angle is at most 180 degrees.

✓Every line segment between two vertices of the polygon does not go exterior to the polygon.

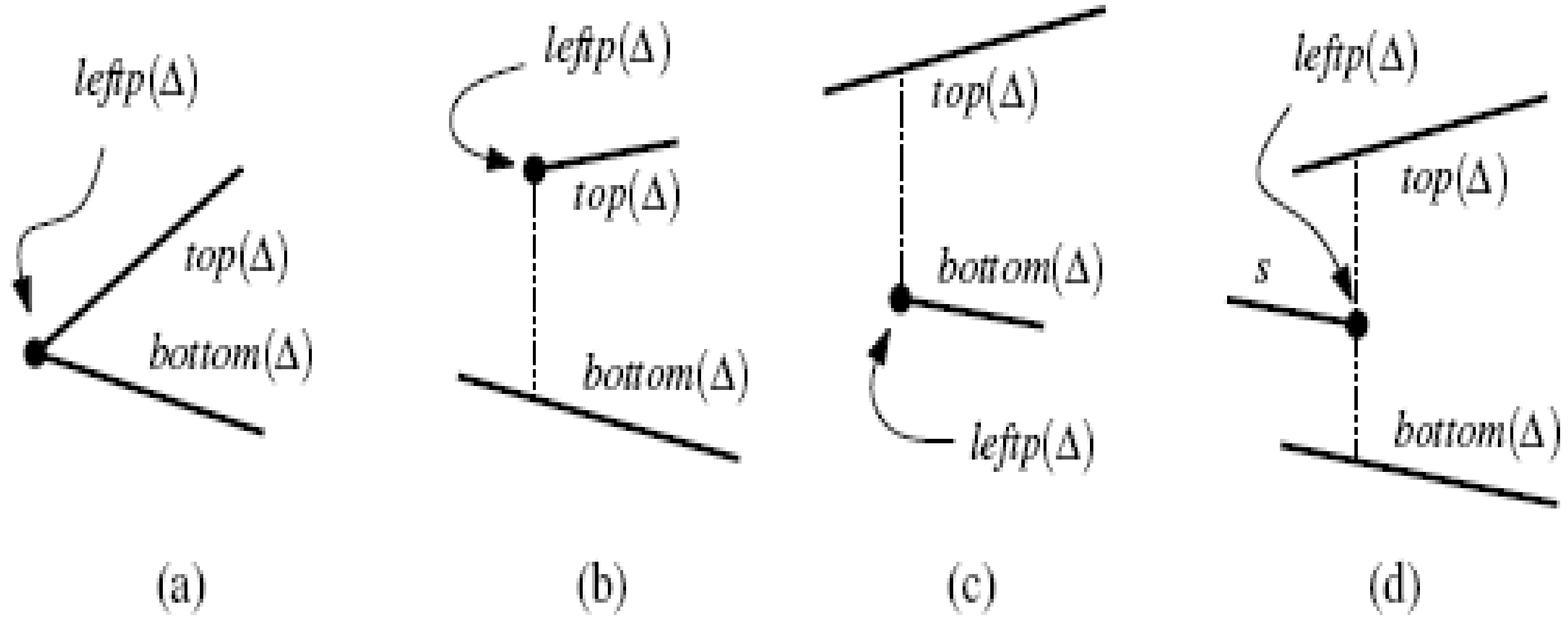


Some notation

Segments top(Δ) and bottom(Δ) :



Points leftp(Δ) and rightp(Δ):



Each Δ is defined by top (Δ), bottom (Δ), leftp(Δ), rightp (Δ).

Lemma 6.2 The trapezoidal map Ƃ(s) of a set s of n line segments in general position contains at most 6n+4 vertical and at most 3n+1 trapezoids.

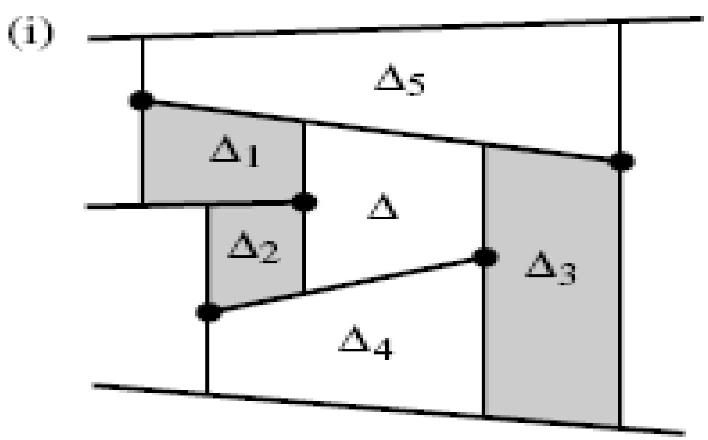
- Two trapezoids are **adjacent** if they share a vertical boundary.

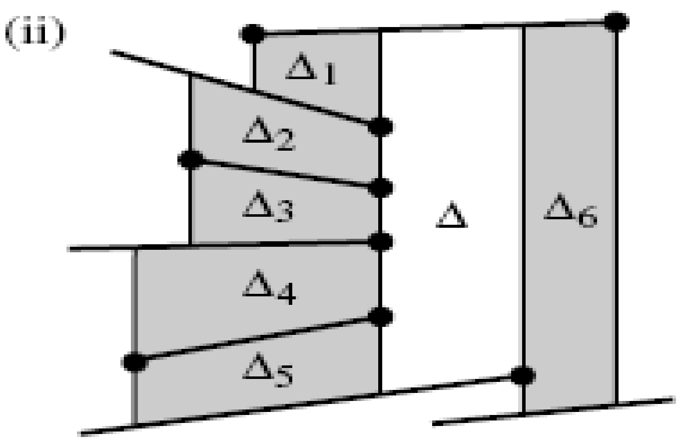
- How many trapezoids can be adjacent to Δ ?

- If Δ' be a trapezoid to Δ along the left vertical edge of Δ and:

▪ Top (Δ') = Top (Δ) then Δ' is upper left neighbor.

▪ Bottom (Δ') = Bottom (Δ) then Δ' is lower left neighbor.



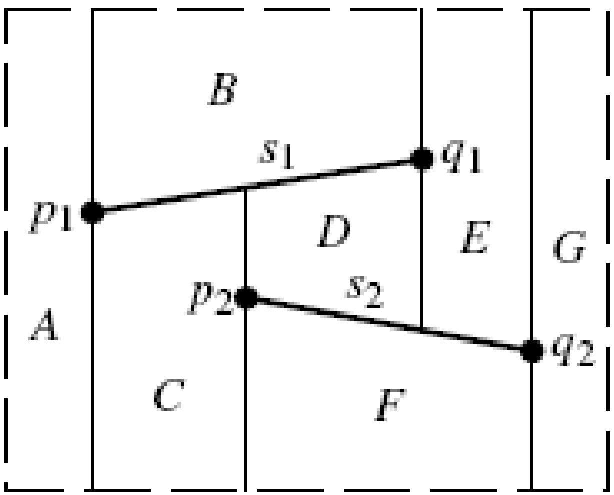


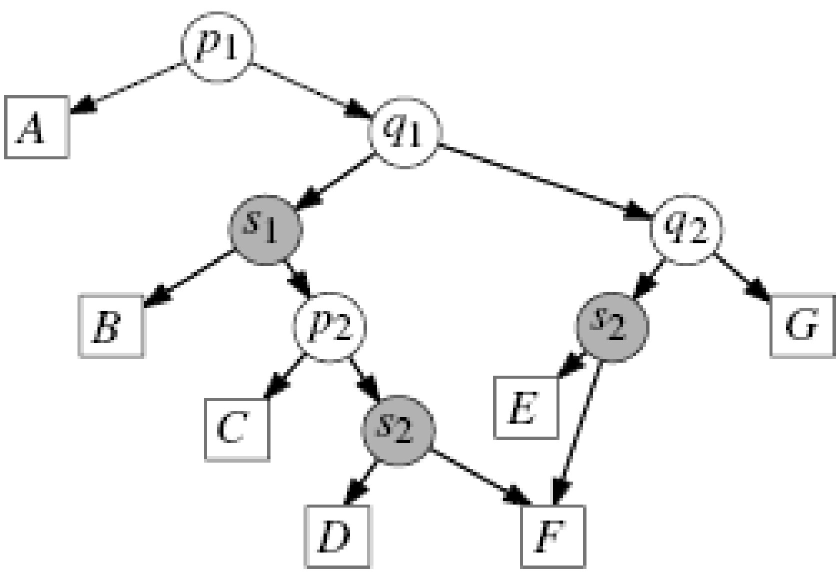
- The record for a trapezoid Δ store pointers to Top(Δ) and Bottom (Δ), pointers to leftp (Δ) and rightp (Δ) and pointers to its at most four neighbors.
- We can deduce the geometry of Δ in constant time from the information stored for Δ.

A Randomized Incremental Algorithm

Search Structure

A directed acyclic graph with a single root and exactly one leaf for every trapezoid of the trapezoid map of S . Its inner node have out-degree 2.There are two types of inner nodes: x-nodes , which are labeled with an endpoint of some segment in S, and y-nodes which are labeled with a segment itself.





Type of tests

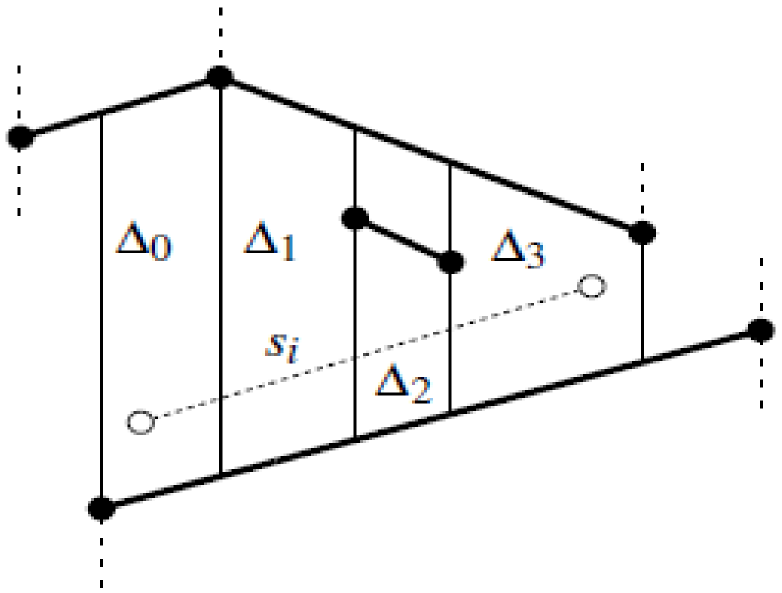
- X-node:

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

- Y-node:

"dose q lie above or below the segment s stored here? "

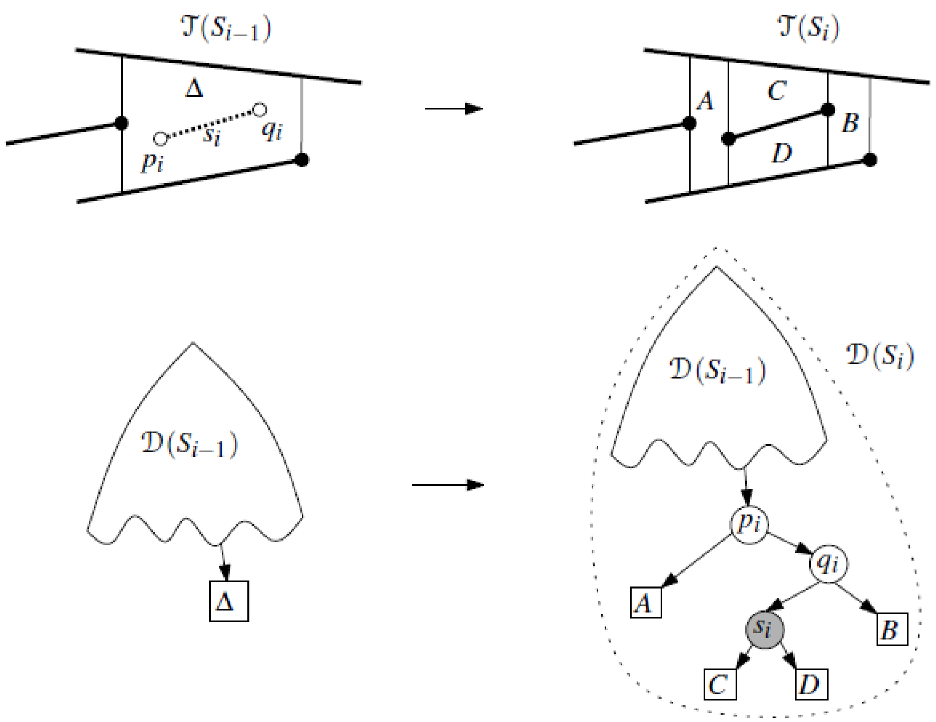
Describe steps



Adding new segment si

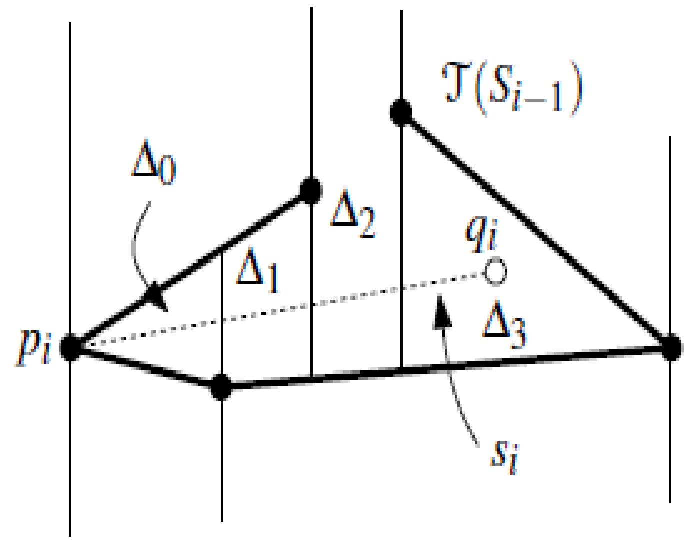
Simple case

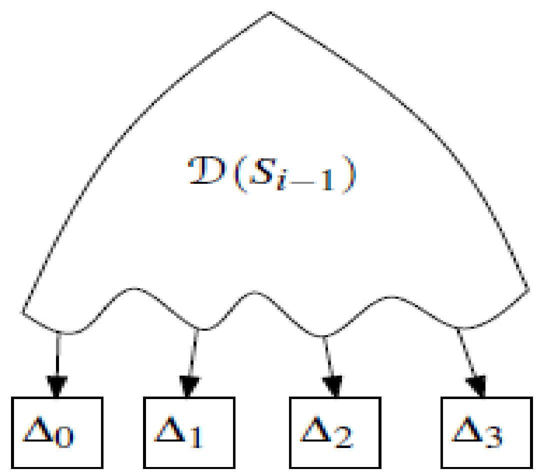
S_i is completely contained in a trapezoid $\Delta = \Delta_0$.



Advanced case

- Let $\Delta_0 \dots \Delta_k$ be the trapezoids intersected by s_i (left to right)
- To find them:
  - Δ_0 is the trapezoid containing the left endpoint p of s_i – find it by querying the data structure built so far.
  - Δ_{j+1} must be a right neighbor of Δ_j .





Updating T

- Draw vertical extensions through the endpoints of si that were not present, partitioning Δ0 … Δk
- Shorten the vertical extensions that now end at si, merging the appropriate trapezoids