Point Location

slides by Andy Mirzaian

(a subset of the original slides are used here)

Planar Point Location:

Knowing where you are on the map

http://www.sciencedirect.com/science/article/pii/S1476927116300378

References:

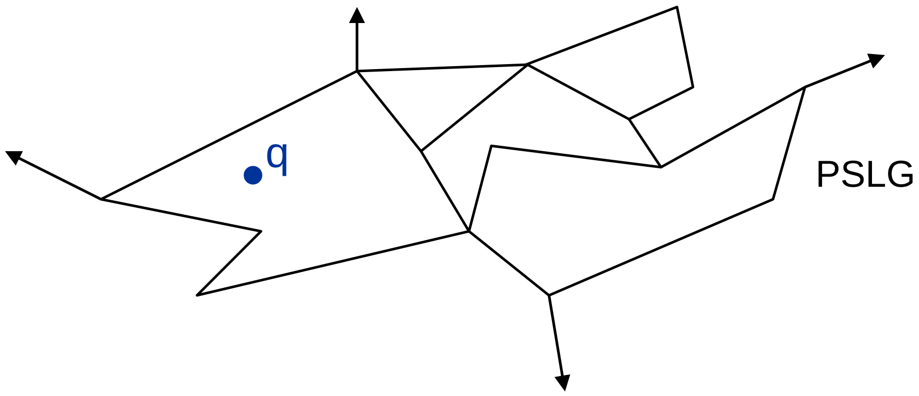
- [M. de Berge et al] chapter 6
- [O'Rourke'98] chapter 7.6
- [Edelsbrunner '87] chapter 11
- [Preparata-Shamos'85] chapter 2.2

Applications:

- GIS: Geographic Information Systems
- Computer Graphics
- Mobile Telecommunication
- Mobile Robotics
- …

Point Location in a Planar Subdivision

PSLG = Planar Straight-Line Graph



Locate a query point q in the PSLG: find which face of the PSLG contains q.

<u>Complexity Measures</u>:

- S - <u>space</u> to store the point location data structure
- T - <u>preprocessing time</u> to construct the data structure
- Q - <u>query time</u> to locate the PSLG face that contains the query point.

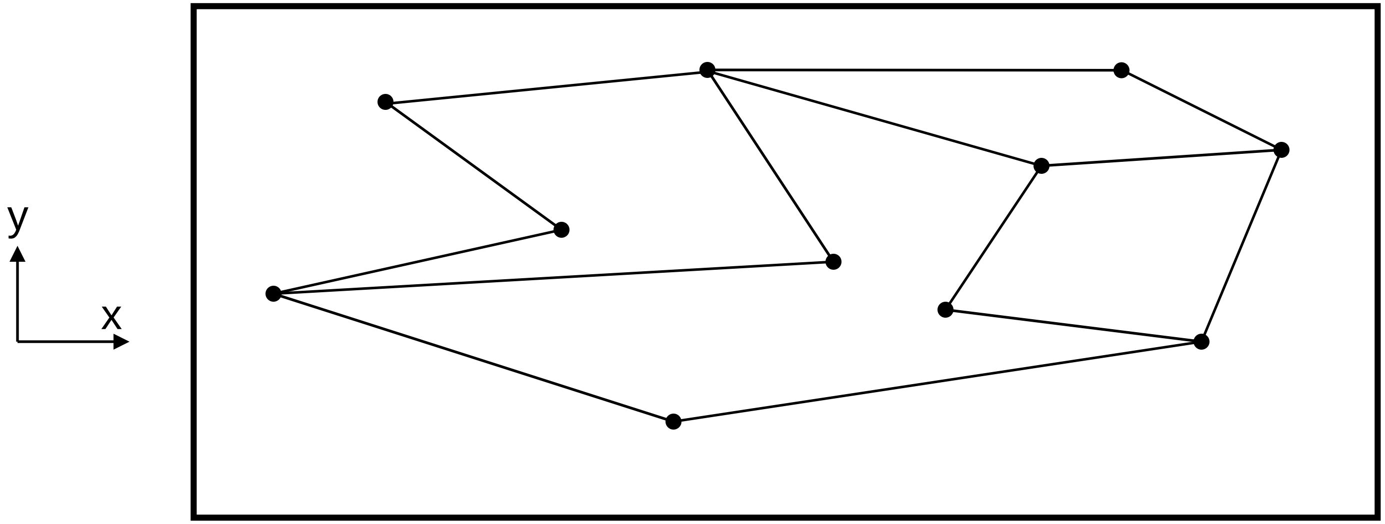
Point Location in a Planar Subdivision

- 1D Optimal method: sorted array, $S = O(n)$, $T = O(n \log n)$, $Q = O(\log n)$.
- 2D: Shamos [1975]: Slab Method: $S = O(n^2)$, $T = O(n^2)$, $Q = O(\log n)$.
- 2D Optimal Method: $S = O(n)$, $T = O(n \log n)$, $Q = O(\log n)$.
  - Mulmuly [1990], Seidel [1991]: Randomized Incremental Method.
  - Kirkpatrick [1983]: Triangulation Refinement Method.
  - Edelsbrunner-Guibas-Stolfi [1986] SIAM J. Computing, pp:317-340.
  - Sarnak-Tarjan [1986], "Planar point location using persistent search trees," Communications of ACM 29, pp: 669-679.
  - Lipton-Tarjan [1977-79]: Planar Separator Method.
- 2D Line Segments intersections:
  Randomized Incremental Method in $O(K + n \log n)$ expected time.

The Slab Method

- O(n^2) space
- O(n^2) preprocessing time
- O(log n) query time.

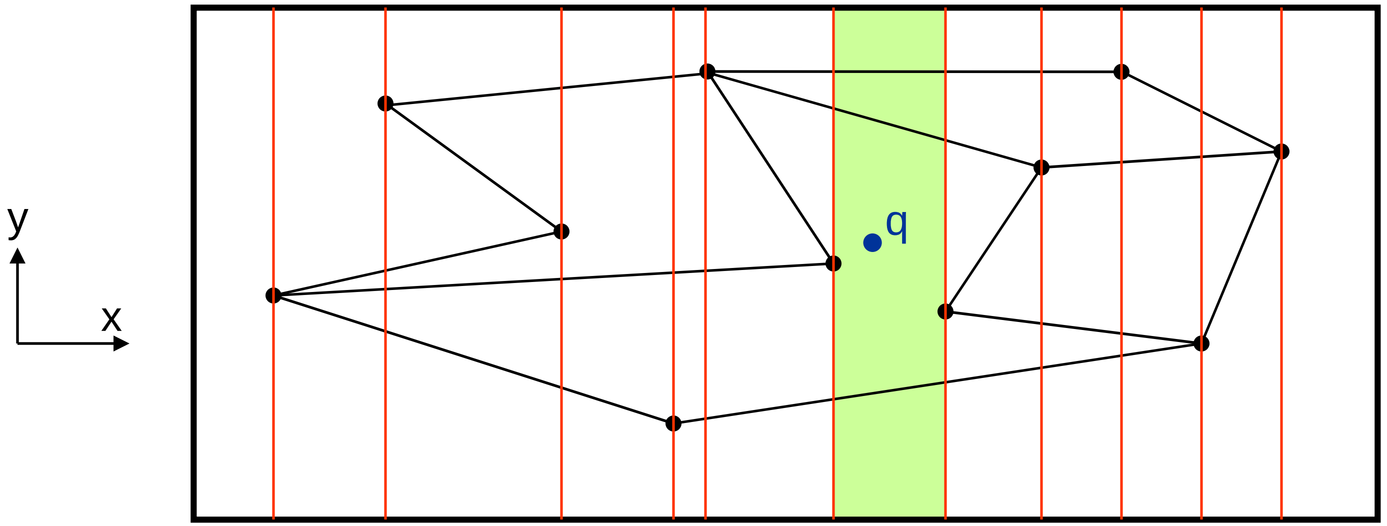
A given PSLG with n vertices (# edges ≤ 3n-6). We may add a large bounding box.



The Slab Method

- ❑ $O(n^2)$ space
- ❑ $O(n^2)$ preprocessing time
- ❑ $O(\log n)$ query time.

A given PSLG with n vertices (# edges ≤ 3n-6). We may add a large bounding box.



a vertical slab

<u>Query Answering</u>:
- do binary search among slabs (in x-sorted order).
- do binary search vertically within the located slab.
- each binary search takes Q = O(log n) time.

Preprocessing for the Slab Method

The Plane Sweep Method:

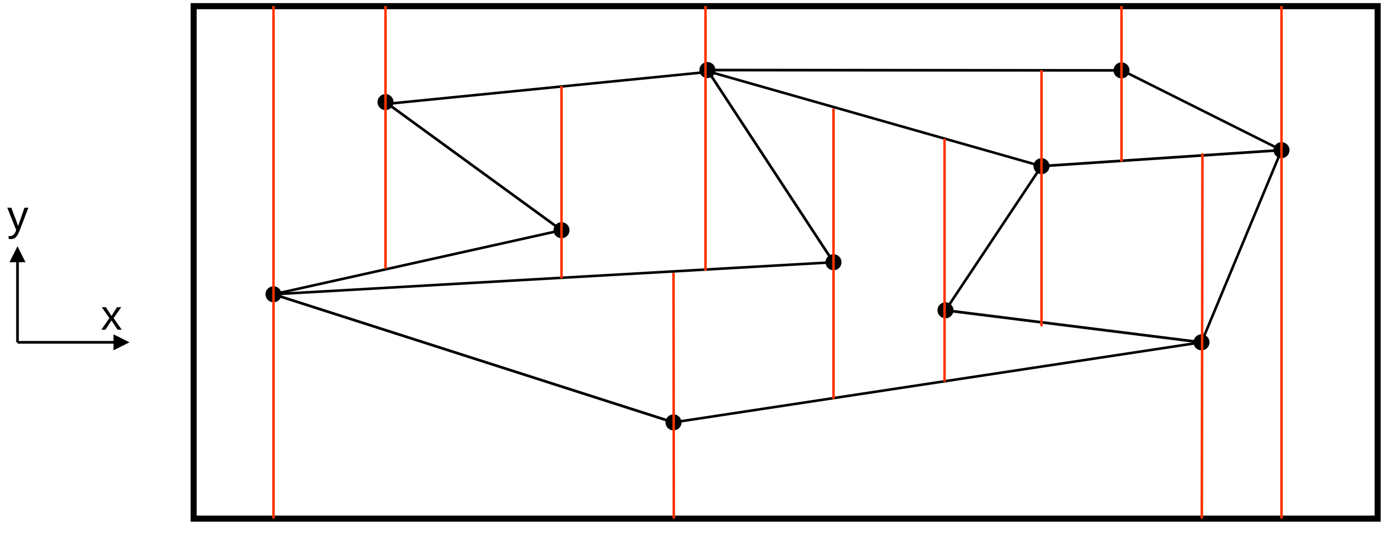
- ❑ Event schedule: x-coordinate of PSLG vertices in increasing order. Maintain these in a priority queue Q.
- ❑ Event Status: vertical sorted ordering of sub-regions within the current slab. Maintain this in a dictionary D.

- ❑ Create a sorted array of slabs. Every time a slab is completed, dump a copy of the current D in the next entry of the sorted array of slabs. [This will be the final data structure.]

- ❑ Analysis:
  - ➢ Event processing takes O(log n) time on Q, O(e_v log n) time on D, and O(n) time to dump a copy of D into the permanent D.S. Here e_v is the number of edges incident to the current event vertex v.
  - ➢ Total Preprocessing Time $T = O(n \log n + \sum_v e_v \log n + n \cdot n)$ $= O(n \log n + n \log n + n^2) = O(n^2)$.
  - ➢ Space = $O(n^2)$.

Randomized Incremental Method

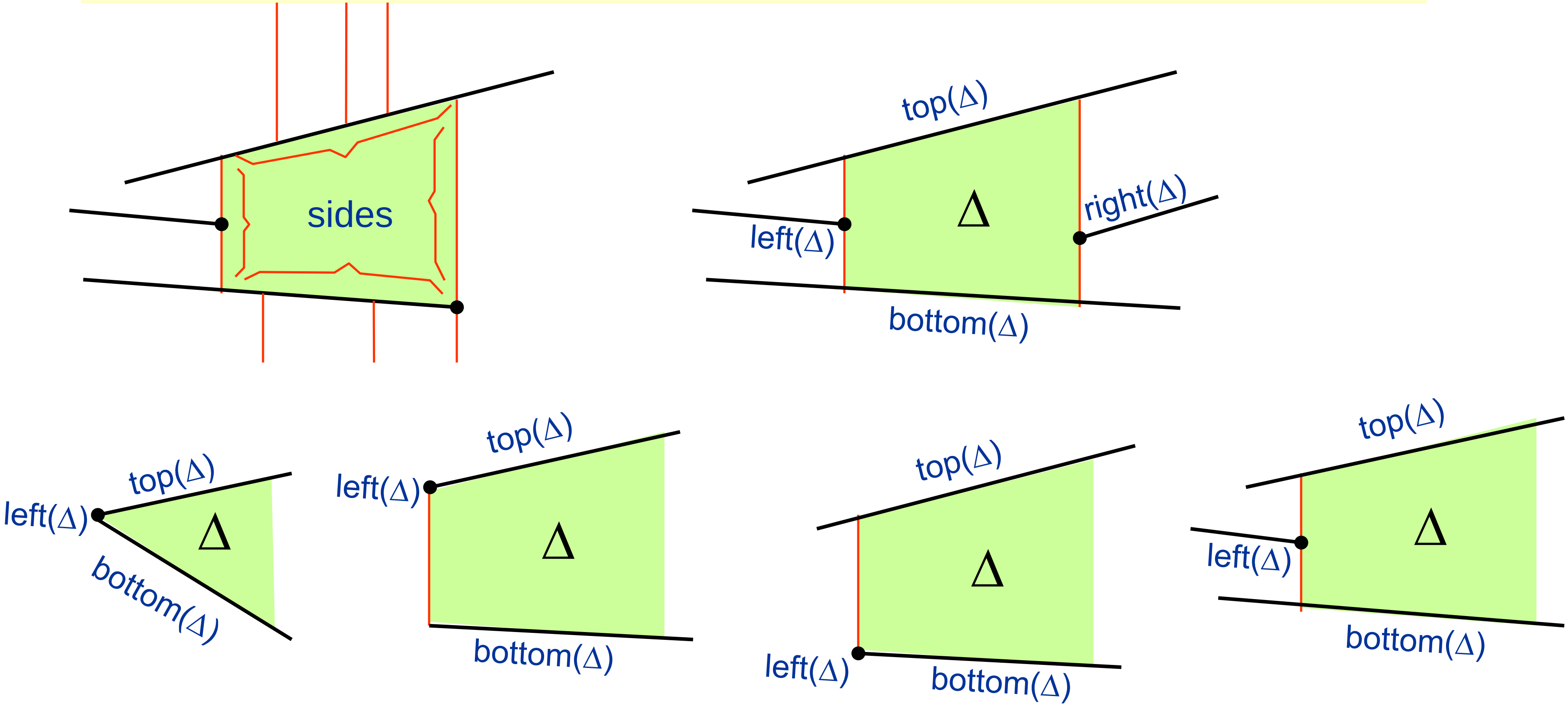
Construct the Trapezoidal decomposition not by the sweep method but by a randomized incremental method. This at the same time constructs the query search structures and also has optimal expected performance.



Randomized Incremental Method

<u>Defining features of a trapezoid Δ</u>:

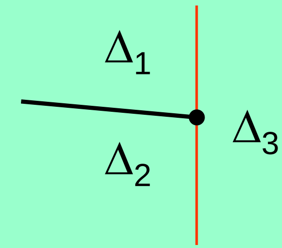
Δ is defined by up to 4 line segments left(Δ), right(Δ), top(Δ), bottom(Δ).
(These are some edges of the PSLG, possibly not all distinct.)



right(Δ) is defined symmetrically.

Randomized Incremental Method

CLAIM: If PSLG has n line segments, then # trapezoids ≤ 3n + 1.



Proof: Assume 2n end-points are in general position.
Each end-point defines left/right wall of at most 3 trapezoids.
Except the leftmost & rightmost trapezoids, each trapezoid is defined by 2 vertical walls (incident to 2 end-points).

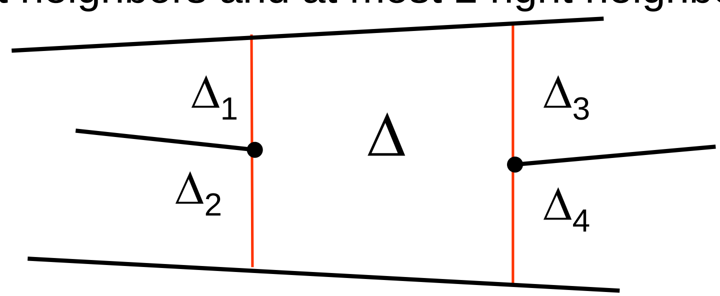
$\therefore$ 2(# trapezoids) – 2 = 3 (# end-points) = 6n.

$\therefore$ # trapezoids = 3n+1.

If end-points are not in general position (i.e., some have equal x-coordinates, or coincide), then the count is even less. [Could use Euler's formula too.]

Trapezoidal Map $\mathcal{T}$(S) O(n) space
of a set S of n non-crossing line segments can be represented by the adjacency structure of its trapezoids.
Adjacency: Δ_1 and Δ_2 are adjacent iff they share (portion of) a vertical wall.

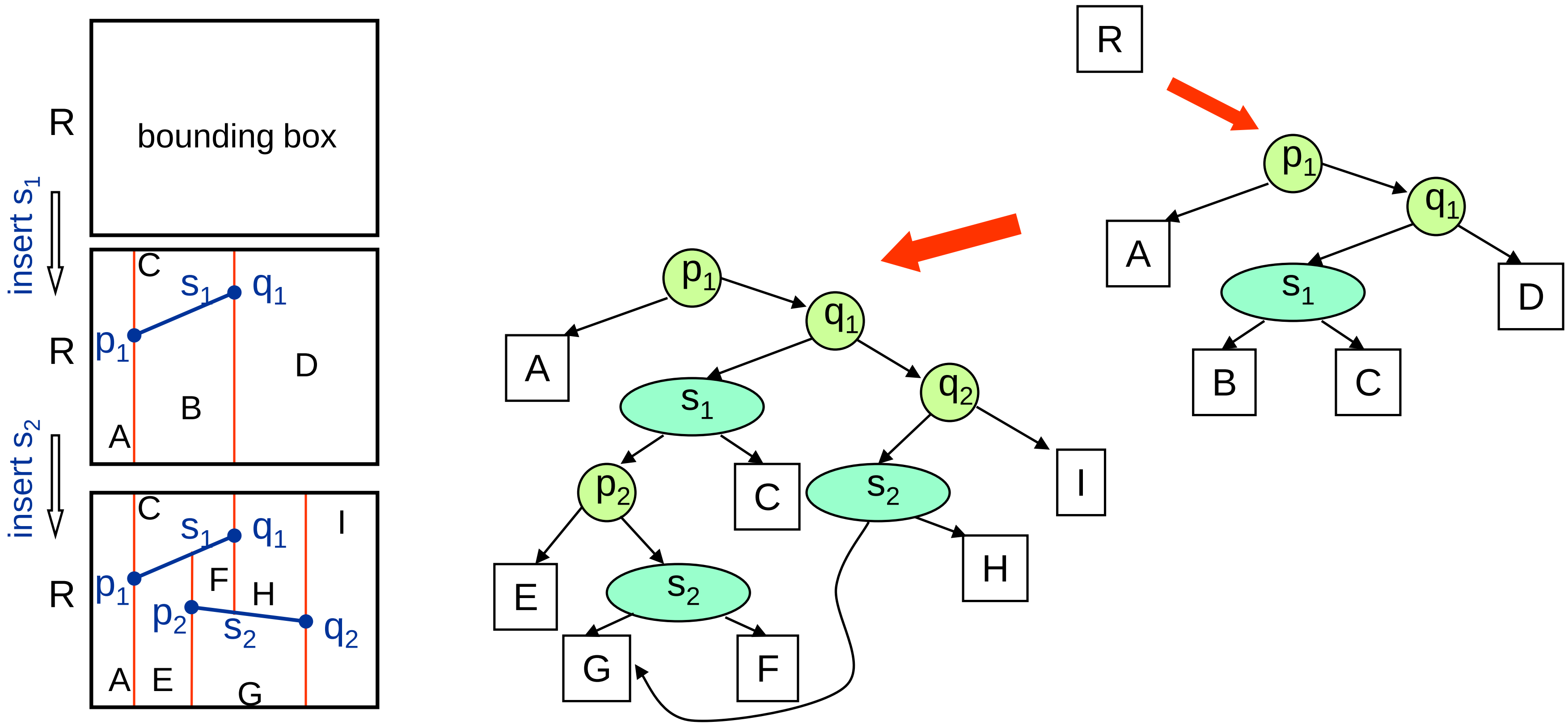
A Δ has at most 2 left neighbors and at most 2 right neighbors.



$\mathcal{D}$(S): The Query Search Structure

- It's a rooted DAG, each node has out-degree at most 2.
- Leaves (i.e., nodes of out-degree 0) store trapezoids with 2-way cross-pointers with their counter-parts in $\mathcal{T}$(S).
- Internal nodes are either endpoints with x-value as key (left/right comparison), or a line-segment of S (below/above comparison).

$\mathcal{T}$ $S = \{ s_1 , s_2 \}$ $\mathcal{D}$



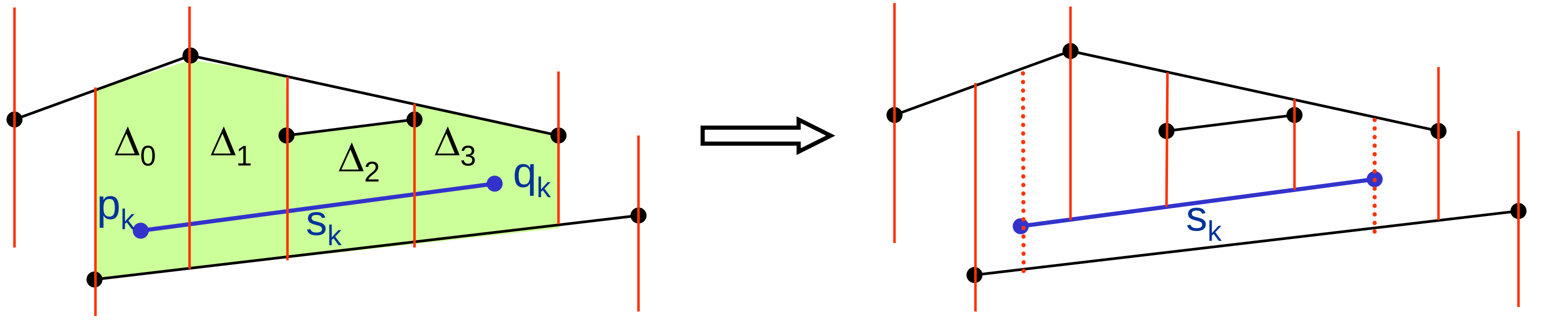
Randomized Incremental Construction of $\mathcal{T}$(S) & $\mathcal{D}$(S)

Input: a set S of n non-crossing line-segments in the plane.
Output: $\mathcal{T}$(S) & $\mathcal{D}$(S).

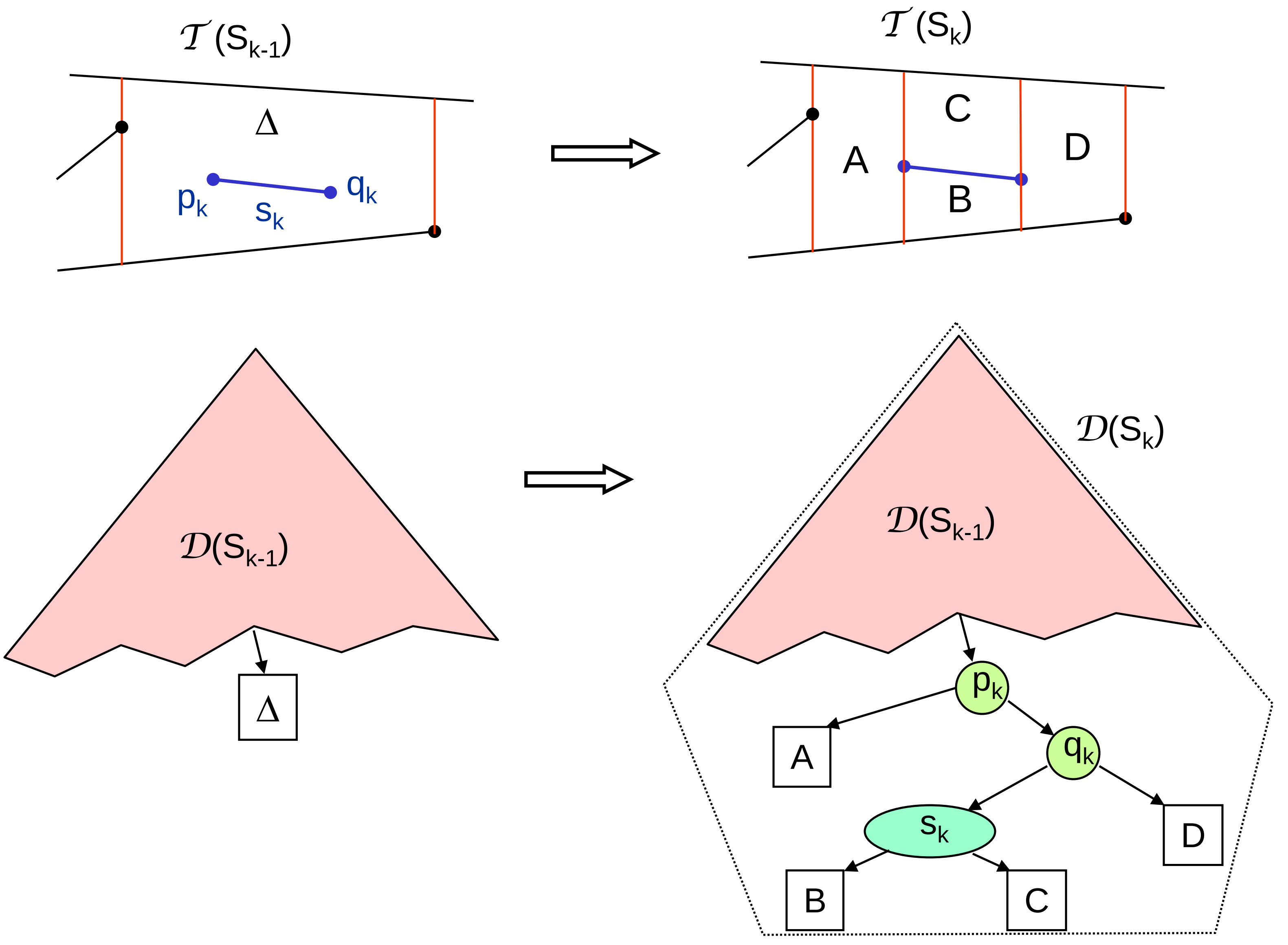
1. Get a bounding box and initialize $\mathcal{T}(\varnothing)$ & $\mathcal{D}(\varnothing)$.
2. Randomly permute S into $(s_1, s_2, \ldots, s_n)$.
3. **for** $k \leftarrow 1..n$ **do**

    (* insert s_k & update $\mathcal{T}(S_k)$ & $\mathcal{D}(S_k)$. $S_k = \{s_1, s_2, \ldots, s_k\}$ *)
    Let p_k & q_k be left & right ends of s_k, respectively
    $\Delta_0 \leftarrow$ Search $(p_k, \mathcal{D})$; $j \leftarrow 0$
    while q_k is to the right of right(Δ_j) **do**
        if s_k is below right(Δ_j)
            then $\Delta_{j+1} \leftarrow$ lower-right-neighbor of Δ_j
            else $\Delta_{j+1} \leftarrow$ upper-right-neighbor of Δ_j
        $j \leftarrow j+1$
    end-while
    $\Delta_0, \Delta_1, \ldots, \Delta_j$ are trapezoids intersected by s_k.
    Update $\mathcal{T}(S_k)$ & $\mathcal{D}(S_k)$ accordingly (see next slide).

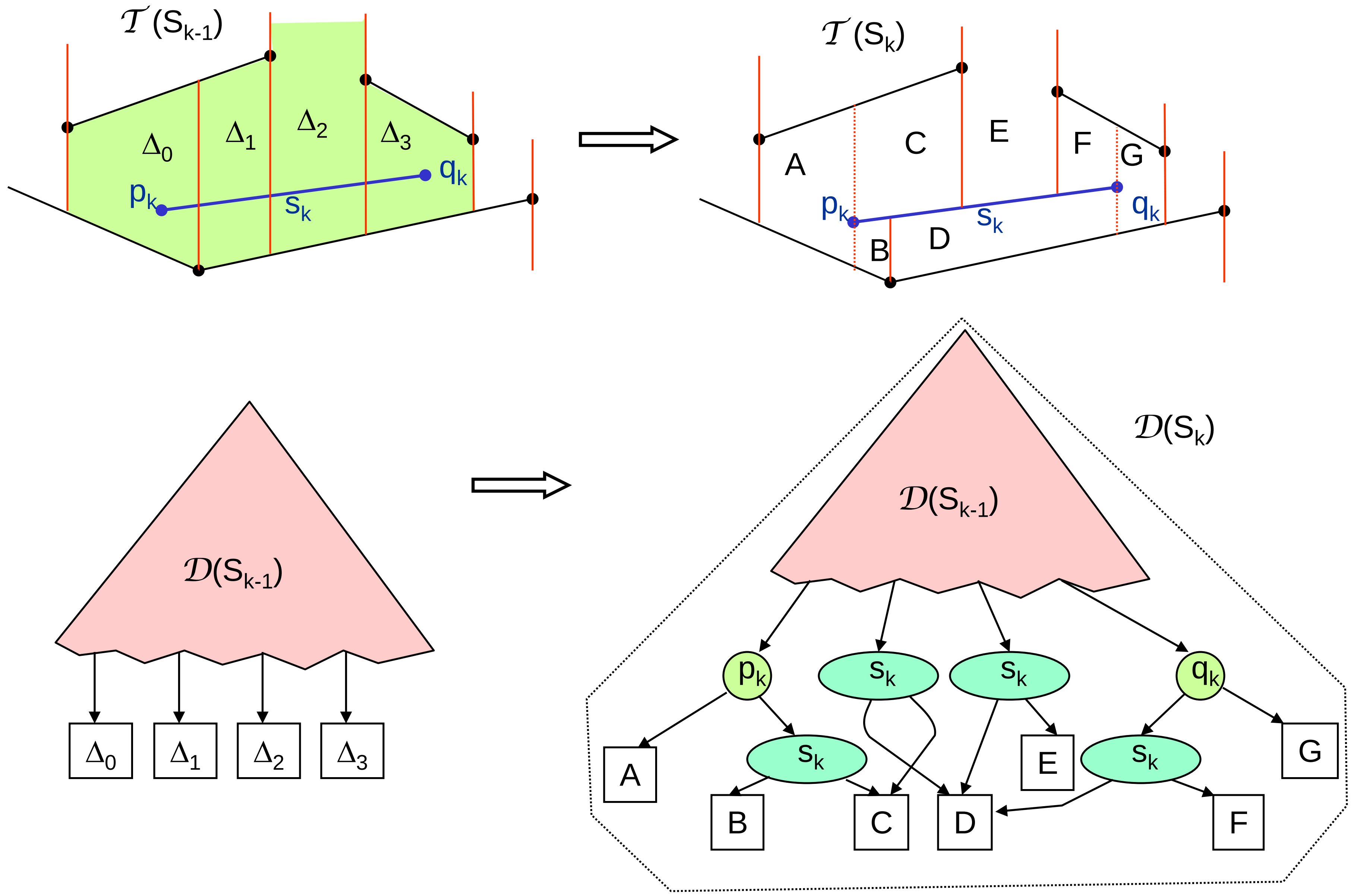
end



Example of step 3



Example of step 3



Complexities

THEOREM: Randomized Incremental algorithm constructs trapezoidal map $\mathcal{T}$(S) & search structure $\mathcal{D}$(S) for a set S of n non-crossing line-segments with complexities:

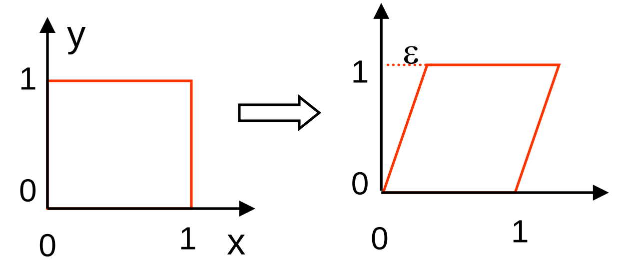
1) O(log n) expected query time for any query point q.
2) O(n) expected size of the search structure.
3) O(n log n) expected construction time.

[All these expectations are on the random ordering of the segments in S.]

Dealing with Degeneracy

What if more than one end-point in S has the same x-coordinate?
How about vertical line-segments in S? …

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



Conceptually assume $\varepsilon > 0$ is sufficiently small.
For a point p = (x,y) assume (x,y) is representing φp = (x+εy , y).

Properties:

1. No two end-points φp & φq of φS have the same (transformed) x-coordinate.

2. Preserves left/right relationships: p left of q $\Leftrightarrow$ φp left of φq.

3. Preserves point-line incidence (it is affine transformation):
   point p above segment s $\Leftrightarrow$ φp above segment φs.

   [Also holds with above replaced by on or below.]