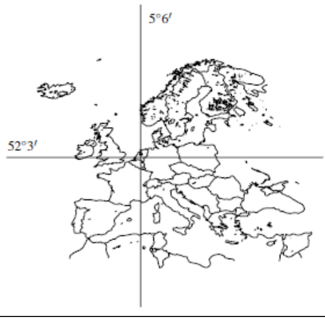
Point Location
Knowing Where You are

Min-Te Sun, Ph.D.

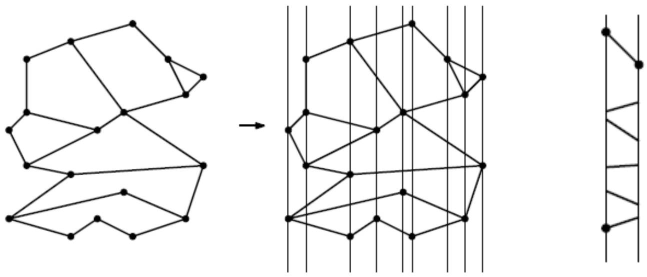
Problem Statement

- Let S be a planar subdivision with n edges. The planar point location problem is to store S in such a way that we can answer queries of the following type:
  - Given a query point q, report the face f of S that contains q.



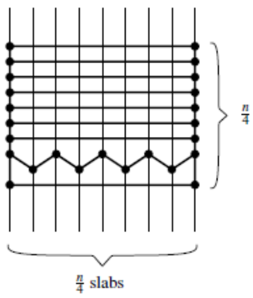
An Intuitive Approach

- Draw vertical lines through all vertices of the subdivision
  - Store x sections in a binary search tree
  - For each x section, store corresponding y sections in another binary tree

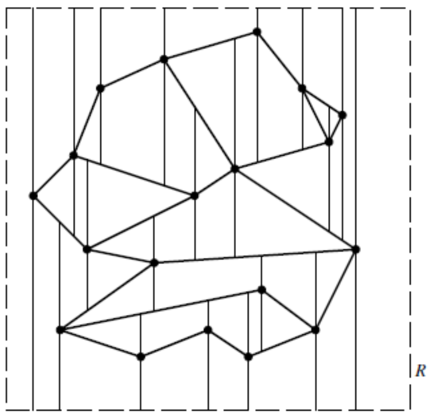


Complexity of Intuitive Approach

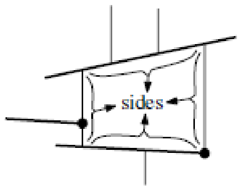
- Time complexity is O(nlogn)
  - Two search operations on binary search trees
- Storage complexity is O(n^2) <= not acceptable!
  - There are O(n) slabs
  - Each slab contains O(n) sections



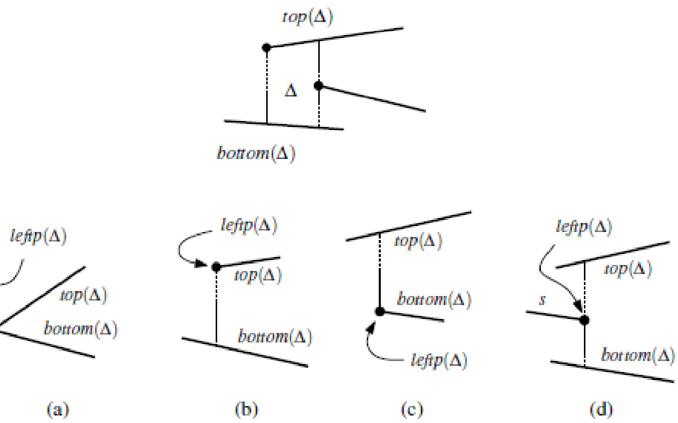
Solution: Trapezoidal Map



- Assumptions: 1) non-crossing; 2) no two distinct endpoints of segments in S have the same x-coordinate (the 2nd one will be removed in section 6.3)
- A trapezoid △ is uniquely defined by the segment top(△), segment bottom(△), left end leftp(△), and right end rightp(△)

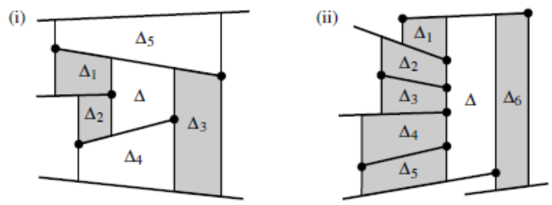


Definitions of Trapezoidal Boundary



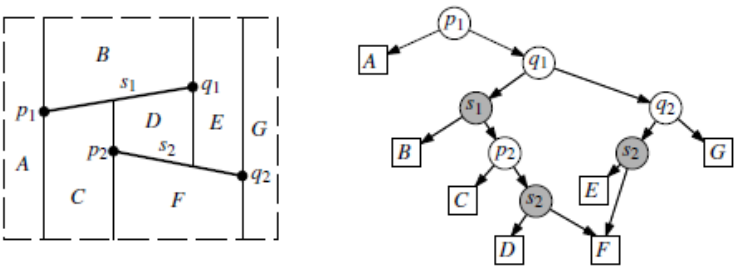
Adjacent Trapezoids

- Two trapezoids Δ and Δ’ are *adjacent* if they meet along a vertical edge.
  - If the set is not in general position, a trapezoid can have an arbitrary number of adjacent trapezoids <= not possible based on our 2nd assumption



Trapezoidial Map Data Structure

- Tree-like graph
  - The internal nodes are the endpoints and segments
  - Each trapezoid is a leaf possibly linked by more than one parent



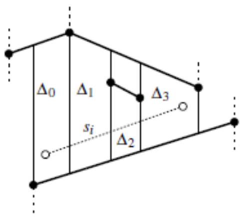
Trapezoidal Map Construction Algorithm

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.

```
Determine a bounding box R that contains all segments of S, and initialize
the trapezoidal map structure T and search structure D for it.
Compute a random permutation s1, s2, ..., sn of the elements of S.
for i ← 1 to n
    do Find the set Δ0, Δ1, ..., Δk of trapezoids in T properly intersected
       by si.
       Remove Δ0, Δ1, ..., Δk from T and replace them by the new trape-
       zoids that appear because of the insertion of si.
       Remove the leaves for Δ0, Δ1, ..., Δk from 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.
```

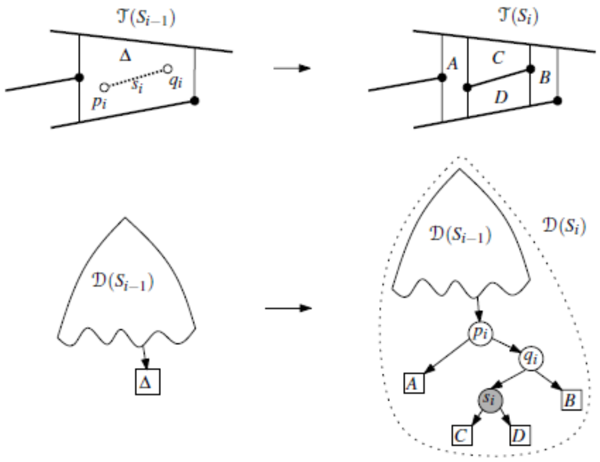
Trapezoids Intersected by s_i

- The trapezoids intersected by s_i must be adjacent to each other
  - To determine which adjacent trapezoid of Δ intersect s_i, just check if rightp(Δ) is above s_i
- We have to find where the endpoint p of s_i is
  1. p is already an endpoint of a S_{i-1}
  2. p is contained in one of the trapezoid Δ in S_{i-1}



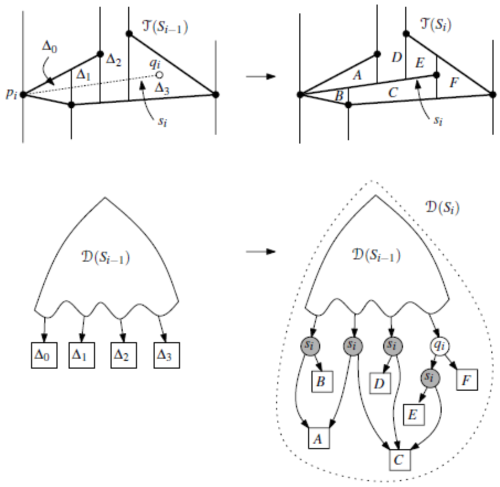
Insertion of s_i : Simple Case

- s_i appears completely inside a trapezoid



Insertion of s_i : Complicated Case

- s_i goes across several trapezoids



of Trapezoids vs # of segments

- The trapezoidal map 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
  - A vertex of T(S) is 1) a vertex of R, 2) an endpoint of a segment, 3) the point where vertical extension starting in an endpoint abuts on another segment => Total # of vertices is bounded by 4+2n+2(2n) = 6n+4
  - leftp(Δ) is either the endpoint of a segment or lower left corner of R. Also, a left endpoint of a segment can be leftp(Δ) for at most 2 trapezoids and a right endpoint of a segment can be leftp(Δ) exactly once => total # of trapezoids is at most 3n+1

Complexity of Trapezoidal Map Construction

- *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* T(*S*) *of a set S of n line segments in general position and a search structure* D *for* 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*).

Query Time Complexity Derivation

- Let X_i, for $1 \leqq I \leqq n$, denote the # of nodes on the search path created in iteration i. the expected path length is

$$\mathrm{E}[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} \mathrm{E}[X_i]$$

- Let P_i denotes the probability that there exists a node on the search path of q that is created in iteration i, we have

$$\mathrm{E}[X_i] \leqslant 3P_i$$

- iteration i contributes a node to the search path of q exactly if $\Delta_q(S_{i-1})$, the trapezoid containing q in $T(S_{i-1})$, is not the same as $\Delta_q(S_i)$, the trapezoid containing q in $T(S_i)$.

$$P_i = \Pr[\Delta_q(S_i) \neq \Delta_q(S_{i-1})]$$

Query Time Complexity Derivation (Cont.)

- When removing s_i will result in the change of $\Delta_q(S_i)$?
  - Only if contributes top, bottom, leftp, or rightp of $\Delta_q(S_i)$

$$P_i = \Pr[\Delta_q(S_i) \neq \Delta_q(S_{i-1})] = \Pr[\Delta_q(S_i) \notin \mathcal{T}(S_{i-1})] \leqslant 4/i.$$

- Using the above formula, we now have

$$\mathrm{E}[\sum_{i=1}^{n} X_i] \leqslant \sum_{i=1}^{n} 3P_i \leqslant \sum_{i=1}^{n} \frac{12}{i} = 12\sum_{i=1}^{n} \frac{1}{i} = 12H_n.$$

- Therefore, the expected query time is O(logn)

Worst Case Storage Complexity Derivation

- Total # of nodes is

$$O(n) + \sum_{i=1}^{n} (\text{number of inner nodes created in iteration } i).$$

- Let k_i be the number of new trapezoids that are created in iteration i, due to the insertion of segment s_i. The number of inner nodes created in iteration i is exactly equal to $k_i - 1$.
  - The worst case upper bound on k_i follows from the fact that the number of new trapezoids in T(S_i) can obviously not be larger than the total number of trapezoids in T(S_i) = $O(i)$. This leads to

$$O(n) + \sum_{i=1}^{n} O(i) = O(n^2).$$

Expected Storage Complexity Derivation

- Similarly, we have

$$O(n) + \mathrm{E}[\sum_{i=1}^{n} (k_i - 1)] = O(n) + \sum_{i=1}^{n} \mathrm{E}[k_i].$$

- For a trapezoid Δ and a segment s, let

$$\delta(\Delta, s) := \begin{cases} 1 & \text{if } \Delta \text{ disappears from } \mathcal{T}(S_i) \text{ when } s \text{ is removed from } S_i, \\ 0 & \text{otherwise.} \end{cases}$$

- There are at most 4 segments that cause a given trapezoid to disappear. Hence,

$$\sum_{s \in S_i} \sum_{\Delta \in \mathcal{T}(S_i)} \delta(\Delta, s) \leqslant 4|\mathcal{T}(S_i)| = O(i).$$

Expected Storage and Construction Time Complexity Derivation

- Using above formula, we have

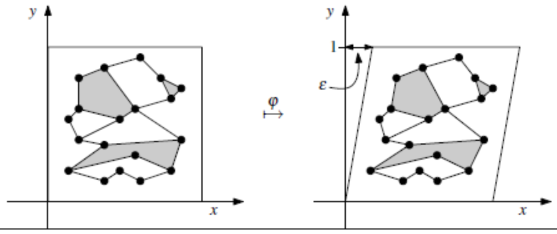
$$\mathrm{E}[k_i] = \frac{1}{i}\sum_{s\in S_i}\sum_{\Delta\in\mathcal{T}(S_i)} \delta(\Delta,s) \leqslant \frac{O(i)}{i} = O(1).$$

- The expected time to insert s_i is $O(k_i)$ plus the time needed to locate the left endpoint of s_i in $T(S_{i-1})$.
  - The expected time complexity for Trapezoidal Map construction is

$$O(1)+\sum_{i=1}^{n}\left\{O(\log i)+O(\mathrm{E}[k_i])\right\} = O(n\log n).$$

Dealing w/ Degenerate Cases

- Degenerate cases
  - Several distinct segments have the same x-coordinate
  - Vertical segments
  - Query point falls on a segment
- Solution: Symbolic Perturbation



Symbolic Perturbation

- Shear transformation: assume some value $\varepsilon > 0$ such that after the following transformation all endpoints have different x-coordinates

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

  - The real ε value is never required at the end ☺
  - Shear transformation preserves
    - the order of points in x-coordinate
    - The “above” or “below” relationship between a point and a segment

Operations Required in Algorithms

- Notice that our algorithms never really need to compute the intersection of lines
- The operations needed in our algorithms
  1. Two points p and q and decides whether q lies to the left, to the right, or on the vertical line through p.
  2. Take one of the input segment, i.e., 2 endpoints p_1 and p_2, and test whether q lies above, below, or on this segment.

Operation One

- For two transformed points $\phi\ p$: $(x_p + \varepsilon y_p, y_p)$ and ϕq: $(x_q + \varepsilon y_q, y_q)$:
  1. If $x_p \neq x_q$ => Trivial
  2. If $x_p = x_q$ and $y_p \neq y_q$ => *use* y_p *and* y_q to determine their relationship
  3. If both $x_p = x_q$ and $y_p = y_q$ => two points are the same!

Operation Two

- Given two endpoints $\phi\ p_1 = (x_1+\varepsilon y_1, y_1)$ and $\phi\ p_2 = (x_2+\varepsilon y_2, y_2)$, we want to test whether a point $\phi q = (x+\varepsilon y, y)$ lies above, below, or on ϕs.
  – Check the vertical line through ϕq, which intersects ϕs. We have

$$x_1 + \varepsilon y_1 \leqslant x + \varepsilon y \leqslant x_2 + \varepsilon y_2.$$

  – If $x = x_1$ then $y \geqq y_1$, and if $x = x_2$ then $y \leqq y_2$.
  – Two cases: $x_1 = x_2$ and $x_1 < x_2$

Construction and Query Problem Transformation

- All endpoints of each segment will have to be transformed “conceptually” (not literally)
- The query point will also need to be transformed
- The face of each trapezoid can be found by looking at the associated face of the top segment

Homework Assignment 3

Page 115

- 5.1
- 5.3
- 5.5

Page 144

- 6.1
- 6.2