Computational Geometry

Chapter 6

Point Location

Problem Definition
- Preprocess a planar map $S$. Given a query point $p$, report the face of $S$ containing $p$.
- Goal: $O(n)$-size data structure that enables $O(\log n)$ query time.
- Application: Which state is Boston located in?
- Trivial Solution: $O(n)$ query time, where $n$ is the complexity of the map. Why?

Naive Solution
- Draw vertical lines through all the vertices of the subdivision.
- Store the $x$-coordinates of the vertices in an ordered binary tree.
- Within each slab, sort the segments separately along $y$.
- Query time: $O(\log n)$.
- Problem: Too delicate subdivision, of size $\Theta(n^2)$ in the worst case.
(Give such an example!)

The Trapezoidal Map
- Construct a bounding box.
- Assume general position: unique $x$ coordinates.
- Extend upward and downward the vertical line from each vertex until it touches another segment.
- This works also for noncrossing line segments.

Properties
- Contains triangles and trapezoids.
- Each trapezoid or triangle is determined:
  - By two vertices that define vertical sides; and
  - By two segments that define nonvertical sides.
- A refinement of the original map.

Notation
- Every trapezoid (triangle) $\Delta$ is defined by
- $\text{Left}(\Delta)$: a segment endpoint (right or left);
- $\text{Right}(\Delta)$: a segment endpoint (right or left);
- $\text{Top}(\Delta)$: a segment;
- $\text{Bottom}(\Delta)$: a segment.
Proof:

1. Vertices:
   \[ 2n + 4n + 4 = 6n + 4 \]
   \[ \uparrow \uparrow \]
   original extensions box

2. Faces: Count Left(Λ).
   \[ 2n + n + 1 = 3n + 1 \]
   \[ \uparrow \uparrow \]
   left e.p. right e.p. box

An alternative:
- The vertices that define its right and left sides;
- The top and bottom segments;
- The (up to two) neighboring trapezoids on right and left;
- (Optional): The neighboring trapezoids from above and below.

This number might be linear in \( n \), so only the information of these trapezoids is stored.

Note: Computing any trapezoidal structure can be done in constant time.

The DAG Search Structure

Query point \( q \), search-structure node \( s \).
- \( s \) is a segment endpoint:
  - \( q \) is to the right of \( s \): go right;
  - \( q \) is to the left of \( s \): go left;
- \( s \) is a segment:
  - \( q \) is below \( s \): go right;
  - \( q \) is above \( s \): go left;

Construction

Find a Bounding Box.
- Randomly permute the segments.
- Insert the segments one by one into the map.
- Update the map and search structure in each insertion.
- The map is independent of the order of insertion and its size is \( O(n) \).
Find in the existing structure the face that contains the left endpoint of the new segment. (*)
Find all other trapezoids intersected by this segment by moving to the right. (In each move choose between two options: Up or Down.)
Update the map $M_i$ and the DAG $D_i$.
(*) Note: Since endpoints may be shared by segments, we need to consider its segment while searching.

The segment is contained entirely in one trapezoid.
- In $M_i$: Split the trapezoid into four trapezoids.
- In $D_i$: The leaf is replaced by a subtree.
- $O(1)$ time.

The $i$th segment intersects $k_i > 1$ trapezoids.
- Split trapezoids.
- Merge trapezoids that can be united.
- $O(k_i)$ time.

Each inner trapezoid in $D_i$ is replaced by:
Each outer trapezoid in $D_i$ is replaced by:

Leaves are eliminated and replaced by one common leaf.

Each segment adds trees of depth at most 3, so the depth of $D_i$ is $\leq 3i$.
Query time (depth of $D_i$):
$O(i), \Theta(i)$ in the worst case.
The $i$th segment — $s_i$ — may intersect with $k_i = O(i)$ trapezoids!
The size of $D$ and its construction time is in the worst case:
$$\sum_{i=1}^{n} \Theta(i) = \Theta(n^2)$$
Segment/Trapezoid Interaction

- One segment may affect many trapezoids
- One trapezoid may affect at most four segments

Average-Case Analysis

Compute the expected depth of $D$:
- $q$: A point, to be searched for in $D$.
- $p$: The probability that a new node was created in the path leading to $q$ in the $i$-th iteration.

Compute $p$ by backward analysis:
- $\Delta_j(M_i)$: The trapezoid containing $q$ in $M_i$.
- Since a new node was created, $\Delta_j(M) \neq \Delta_j(M_i)$.
- Delete $q_i$ from $M_i$.
- $\text{Prob}(\Delta_j(M) \neq \Delta_j(M_i)) \leq \frac{4}{i}$.

Expected Depth of $D$

- $x_i$: The number of nodes created in the $i$-th iteration in the path leading to the leaf $q$.
- The expected length of the path leading to $q$:
  \[
  E\left[\sum_{i=0}^{n} x_i\right] = \sum_{i=0}^{n} E[x_i] \leq \sum_{i=0}^{n} (3p_i) \leq \sum_{i=0}^{n} \frac{4}{i} = O(\log n).
  \]

Expected Size of $D$

- Define an indicator
  \[\delta(\Delta, s) = \begin{cases} 1 & \Delta \text{ disappears from } M_i \text{ if } s \text{ is removed} \\ 0 & \text{otherwise} \end{cases}\]
- $k_i$: Number of leaves created in the $i$-th iteration.
- $S_i$: The set of the first $i$ segments.
- Average on $s$:
  \[
  E[k_i] = \sum_{i=0}^{n} \left( \sum_{s \in S_i} \delta(\Delta, s) \right) = \sum_{i=0}^{n} \left( \sum_{s \in S_i} \delta(\Delta, s) \right) 
  \leq \sum_{i=0}^{n} \frac{4}{i} 
  = \frac{4}{n} = O(1).
  \]

Expected Size of $D$ (cont.)

- $k_i$: Number of internal nodes created in the $i$-th step.
- Total size:
  \[
  O(n) + E\left[\sum_{i=0}^{n} (k_i - 1)\right] = O(n) + E\left[\sum_{i=0}^{n} k_i\right] = O(n).
  \]

Expected Construction Time of $D$

\[
\sum_{i=1}^{n} (O(\log i) + O(E[k_i])) = O(n \log n)
\]

Finding the first trapezoid

The rest of the work in the $i$-th step
Handling Degeneracies

- What happens if two segment endpoints have the same x coordinate?
- Use a shear transformation:
  \[
  \begin{pmatrix}
  x \\
  y
  \end{pmatrix}
  \rightarrow
  \begin{pmatrix}
  x + \varepsilon y \\
  y
  \end{pmatrix}
  \]
  - Higher points will move to the right.
  - \( \varepsilon \) should be small enough so that this transform will not change the order of two points with different x coordinates.
  - In fact, no need to explicitly shear the plane. Comparison rules can mimic the shearing.
- Prove: The entire algorithm remains correct.