**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?

---

**Naïve 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:
  - $\text{Left}(\Delta)$: The left segment of $\Delta$.
    - (actually it is enough (will see later why) only an endpoint intersecting by the segment.
    - It is either right endpoint or a left endpoint;
  - $\text{Right}(\Delta)$: a segment endpoint (right or left);
  - $\text{Top}(\Delta)$: a segment;
  - $\text{Bottom}(\Delta)$: a segment.
Complexity

- **Theorem** (linear complexity): A trapezoidal map of \( n \) segments contains at most \( 6n + 4 \) vertices and at most \( 3n + 1 \) faces.

- **Proof:**
  1. Vertices:
     \[
     2n + 4n + 4 = 6n + 4
     \]
     original extensions box
  2. Faces: Count Left
     \[
     2n + n + 1 = 3n + 1
     \]
     left \( \rightarrow \) right \( \rightarrow \) box

Map Data Structure

- Possibly by DCEL.

- An alternative:
  For each trapezoid store:
  1. The vertices that define its right and left sides;
  2. The top and bottom segments;
  3. The (up to two) neighboring trapezoids on right and left;
  4. (Optional) The neighboring trapezoids from above and below.

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

DAG

A Directed Acyclic Graph (in our setting) has a unique root and a path connecting the root to each node.

The DAG Search Structure

Given a query point \( q \) how can we find the trapezoid containing \( q \)?

Assume a search-structure node \( s \) is given (initially the root of the DAG)

Search(\( q, s \)):

1. \( ^{\uparrow} \) Query point \( q \), search-structure node \( s \).
2. If \( s \) is a segment-endpoint then
   a. \( q \) is to the right of \( s \): go right;
   b. \( q \) is to the left of \( s \): go left;
   c. **No use of the \( y \)-coordinates of \( s \)**
   d. Else:
3. If \( s \) is a segment:
   a. \( q \) is below \( s \): go right;
   b. \( q \) is above \( s \): go left;
Construction Algorithm (high level description)

- Input: A set of segments
- Output: the DAG

1. Find a Bounding Box.
2. Randomly permute the segments.
3. Insert the segments one by one into the map.
4. Update the map and search structure in each insertion.
5. The map is independent of the order of insertion and its size is \( \Theta(n) \).
6. The DAG and its size depends on the order of insertion.

Update D: Simple Case

The segment is contained entirely in one trapezoid.

- In \( M \): Split the trapezoid into four trapezoids.
- In \( D \): The leaf is replaced by a subtree.
- \( O(1) \) time.

Update M: General Case

The \( p \) segment intersects \( k \geq 1 \) trapezoids.

- Split trapezoids.
- Merge trapezoids that can be united.
- \( O(k) \) time.

Updating D: Split

- Each inner trapezoid in \( D \) is replaced by:
  - A
  - B

- Each outer trapezoid in \( D \) is replaced by:
  - A
  - B
  - C

Updating D: Merge

- Leaves are eliminated and replaced by one common leaf.
- \( O(k) \) time.
Size of $D$: Worst-Case Analysis

1. Each segment adds trees of depth at most 3, so the depth of $D_i$ is $\leq 3i$.
2. Query time (depth of $D$): $O(i)$, $\Theta(i)$ in the worst case.
3. The $s_i$ segment – $s_i$ may intersect with $k_i = O(i)$ trapezoids!
4. The size of $D$ and its construction time is in the worst case:
   $$\sum \Theta(i) = \Theta(n^2)$$

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 $D$.
- $p_i$: The probability that a new node was created in the path leading to $q$ in the $i$th iteration.

Compute $p_i$ by backward analysis:

- $\Delta_q(M_i)$: The trapezoid containing $q$ in $M_i$.
- Since a new node was created, $\Delta_q(M) = \Delta_q(M_i)$.
- Delete $s_i$ from $M_i$.
- $\text{Prob}[\Delta_q(M) \neq \Delta_q(M_i)] \leq 4/i$.

The expected length of the path leading to $q$:

$$E\left[\sum x_i\right] = \sum E[x_i] \leq \sum \left(3p_i\right) \leq \sum \frac{4}{i} = O(\log n).$$

Expected Depth of $D$

- $x_i$: The number of nodes created in the $i$th iteration in the path leading to the leaf $q$.

Theorems

- The expected size is $O(n)$
- The expected query time is $O(\log n)$

(These proofs from this point are not required, but we will handwave a bit.)
**Expected Size of D**

- Define an indicator:
  \[ \delta(\Delta, s) = \begin{cases} 
1 & \text{if } \Delta \text{ disappears from } M_j \text{ if } s \text{ is removed} \\
0 & \text{otherwise}
\end{cases} \]

- \( k_i \): Number of leaves created in the \( i \)th iteration.
- \( S_i \): The set of the first \( i \) segments.

**Average on \( s \):**

\[ \bar{\Delta}(s) = \frac{1}{i} \left( \sum_{s} \delta(\Delta, s) \right) = \frac{i}{4} \sum_{s} \delta(\Delta, s) \]

\[ = \frac{\Delta}{4} \cdot M_j \]

\[ = \frac{\Delta}{4} = O(1). \]

**Expected Construction Time of D**

\[ \sum_{i=1}^{\infty} \left( O(\log i) + O(E[k_j]) \right) = O(n \log n) \]

- Finding the first trapezoid
- The rest of the work in the \( i \)th step

---

**Expected Size of D (cont.)**

- \( k_{i-1} \): Number of internal nodes created in the \( i \)th step.

**Total size:**

\[ O(n) + E \left( \sum_{i} (k_i - 1) \right) = O(n) + E \left( \sum k_i \right) = O(n). \]