Computational Geometry

Winter term 2016/17

Point Localization

or

Where am I?

Lecture #6

Joachim Spoerhase

Chair for Computer Science I
What’s the Problem?

**Task:** Given a planar subdivision $S$ with $n$ segments, preprocess $S$ to allow for fast point location queries!

**Solution:** Pre-proc: Partition $S$ into slabs induced by vertices.

Query: $\begin{align*}
&\text{– find right slab} \\
&\text{– search slab}
\end{align*}$ $\{\text{2 bin. searches!} \quad O(\log n)\}$ time!

**But:** Space? $\Theta(n^2)$ Pre-proc? $O(n^2 \log n)$
Decreasing the Complexity

Observation: The slab partition of $S$ is a refinement $S'$ of $S$ that consists of (possibly degenerate) trapezoids.

Task: Find “good” refinement of $S$ of low complexity!

Solution: Trapezoidal map $\mathcal{T}(S)$
Decreasing the Complexity

**Observation:** The slab partition of $S$ is a *refinement* $S'$ of $S$ that consists of (possibly degenerate) trapezoids.

**Task:** Find “good” refinement of $S$ of low complexity!

**Solution:** *Trapezoidal map $T(S)$*

**Assumption:** $S$ is in *general position*, that is, no two vertices have the same $x$-coordinates.
Notation

**Definition:** A *side* of a face of $\mathcal{T}(S)$ is a segment of maximum length contained in the boundary of the face.
Notation

Definition: A side of a face of $\mathcal{T}(\mathcal{S})$ is a segment of maximum length contained in the boundary of the face.

Observation: $\mathcal{S}$ in gen. pos. $\Rightarrow$ each face $\Delta$ of $\mathcal{T}(\mathcal{S})$ has:
- one or two vertical sides
- exactly 2 non-vertical sides

Left side: $\text{leftp}(\Delta)$
Complexity of $\mathcal{T}(S)$

**Observe:** A face $\Delta$ of $\mathcal{T}(S)$ is uniquely defined by $\text{top}(\Delta), \text{bot}(\Delta), \text{leftp}(\Delta)$, and $\text{rightp}(\Delta)$.

**Lemma.** $S$ planar subdivision in gen. pos., with $n$ segments $\Rightarrow \mathcal{T}(S)$ has $\leq 6n + 4$ vtc and $\leq 3n + 1$ trapezoids.

**Proof.** The vertices of $\mathcal{T}(S)$ are

- endpts of segments in $S$ $\leq 2n$
- endpts of vertical extensions $\leq 2 \cdot 2n$
- vertices of $R$ $\leq 4$

Bound $\#\text{trapezoids}$ via Euler or directly (segments/leftp).

**Approach:** Construct tapezoidal map $\mathcal{T}(S)$ and point-location data structure $\mathcal{D}(S)$ for $\mathcal{T}(S)$ *incrementally*!

algorithm-design paradigm!
The 1d-Problem

Given a set $S$ of $n$ real numbers...

$S_{i-1} := \{s_1, \ldots, s_{i-1}\}$, $l_{i-1} := \text{set of conn. comp. of } \mathbb{R} \setminus S_{i-1}$

- pick an arbitrary point $s_i$ from $S \setminus S_{i-1}$
- locate $s_i$ in the search structure $D_{i-1}$ of $S_{i-1}$
- split interval $(\ell, r)$ of $l_{i-1}$ containing $s_i$
- build $D_i$:

Problem: looong search paths!
The 1d-Problem

Given a set $S$ of $n$ real numbers...

$S_{i-1} := \{s_1, \ldots, s_{i-1}\}$, $I_{i-1} := \text{set of conn. comp. of } \mathbb{R} \setminus S_{i-1}$

**Solution:** *random!*

- pick an arbitrary point $s_i$ from $S \setminus S_{i-1}$
- locate $s_i$ in the search structure $D_{i-1}$ of $S_{i-1}$
- split interval $(\ell, r)$ of $I_{i-1}$ containing $s_i$
- build $D_i$:

**Problem:** *loooong search paths!*
1d Result

Given a set $S$ of $n$ real numbers...

$S_{i-1} := \{s_1, \ldots, s_{i-1}\}$, \hspace{1cm} $l_{i-1} := \text{set of conn. comp. of } \mathbb{R} \setminus S_{i-1}$

**Thm.** The randomized-incremental algorithm preprocesses a set $S$ of $n$ reals in $O(n \log n)$ expected time such that a query takes $O(\log n)$ expected time.

**Proof.** Let $q \in \mathbb{R}$ (wlog. $q \not\in S$) and $l_i(q) = \arg\{l \in l_i : q \in l\}$.

Define random variable $X_i = \begin{cases} 1 & \text{if } l_i(q) \neq l_{i-1}(q), \\ 0 & \text{else}. \end{cases}$

$$E[\text{query time in } \mathcal{D}_n] = E[\text{length search path in } \mathcal{D}_n] =$$

$$= E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i] = ?$$
Expected Query Time of $\mathcal{D}_n$

Define random variable $X_i = \begin{cases} 1 & \text{if } l_i(q) \neq l_{i-1}(q), \text{ i.e., } s_i \in l_{i-1}(q), \\ 0 & \text{else.} \end{cases}$

$E[X_i] = P[X_i = 1] = \frac{2}{i}$

$= \text{probability that } l_i(q) \neq l_{i-1}(q), \text{ i.e., } s_i \in l_{i-1}(q)$.

**Backwards analysis:** Consider $S_i$ fixed. If we remove a randomly chosen pt from $S_i$, what’s the probability that the interval containing $q$ changes?
– we have $i$ choices, identically distributed
– at most two of these change the interval

Define random variable $X_i = \begin{cases} 1 & \text{if } l_i(q) \neq l_{i-1}(q), \\ 0 & \text{else.} \end{cases}$

$E[\text{query time in } \mathcal{D}_n] = E[\text{length search path in } \mathcal{D}_n] = E[\sum_{i=1}^n X_i] = \sum_{i=1}^n E[X_i] = \mathcal{O}(\log n)$
The 1d-Result

**Thm.** The randomized-incremental algorithm preprocesses a set $S$ of $n$ reals in $O(n \log n)$ expected time such that a query takes $O(\log n)$ expected time.
The 2d-Problem

**Approach:** randomized-incremental construction of $\mathcal{T}$ and $\mathcal{D}$
- use $\mathcal{D}$ to locate left endpoint of next segment $s$
- "walk" along $s$ through $\mathcal{T}$
- destroy all trapezoids of $\mathcal{T}$ intersecting $s$
- construct new trapezoids of $\mathcal{T}$ (adjacent to $s$)
- update $\mathcal{D}$
Walking through $\mathcal{T}$ and Updating $\mathcal{D}$

\[
\text{TrapezoidalMap(set } S \text{ of } n \text{ non-crossing segments)}
\]

\[
R = \text{BBox}(S); \quad \mathcal{T}.\text{init}(); \quad \mathcal{D}.\text{init}()
\]

\[
(s_1, s_2, \ldots, s_n) = \text{RandomPermutation}(S)
\]

\[\text{for } i = 1 \text{ to } n \text{ do}\]

\[ (\Delta_0, \ldots, \Delta_k) = \text{FollowSegment}(\mathcal{T}, \mathcal{D}, s_i) \]

\[ \mathcal{T}.\text{remove}(\Delta_0, \ldots, \Delta_k) \]

\[ \mathcal{T}.\text{add(new trapezoids incident to } s_i) \]

\[ \mathcal{D}.\text{remove}_\text{leaves}(\Delta_0, \ldots, \Delta_k) \]

\[ \mathcal{D}.\text{add}_\text{leaves(new trapezoids incident to } s_i) \]

\[ \mathcal{D}.\text{add}_\text{new}_\text{inner}_\text{nodes}() \]
The 2d-Result

**Theorem.** TrapezoidalMap(S) computes $\mathcal{T}(S)$ for a set of $n$ line segments in general position and a search structure $\mathcal{D}$ for $\mathcal{T}(S)$ in $O(n \log n)$ expected time. The expected size of $\mathcal{D}$ is $O(n)$ and the expected query time is $O(\log n)$.

**Invariant:** Before step $i$, $\mathcal{T}$ is a trapezoidal map for $S_{i-1}$ and $\mathcal{D}$ is a valid search structure for $\mathcal{T}$.

**Proof.**
- Correctness by loop invariant.
- Query time similar to 1d analysis.
  $\Rightarrow$ construction time
Query Time

Let \( T(q) \) be the query time for a fixed query pt \( q \).
\[ \Rightarrow T(q) = O(\text{length of the path from } D.\text{root to } q). \]

height(\( D \)) increases by at most 3 in each step. \( \Rightarrow T(q) \leq 3n. \)

We are interested in the expected behaviour of \( D \):
\[ \Rightarrow \text{average of } T(q) \text{ over all } n! \text{ insertion orders (permut. of } S) \]

\( X_i := \# \text{ nodes that are added to the query path in iteration } i. \)
\( S \) and \( q \) are fixed.
\[ \Rightarrow X_i \text{ random variable that depends only on insertion order of } S. \]
\[ \Rightarrow \text{expected path length from } D.\text{root to } q \text{ is} \]
\[ E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i] = ? \]
Query Time (cont’d)

\( p_i \) = prob. that the search path \( \Pi_q \) of \( q \) in \( \mathcal{D} \) contains a node that was created in iteration \( i \).

\[ \Rightarrow E[X_i] = \sum_{j=0}^{3} j \cdot P[X_i = j] \leq \sum_{j=0}^{3} 3 \cdot P[X_i \geq 1] = 3p_i \]

\( \Delta_q(S_i) := \) trapezoid in \( \mathcal{T}(S_i) \) that contains \( q \).

**Key idea:** Iteration \( i \) contributes a node to \( \Pi_q \) iff

\[ \Delta_q(S_{i-1}) \neq \Delta_q(S_i). \]

In this case \( \Delta_q(S_i) \) must have been created in iteration \( i \).

\[ \Rightarrow \Delta := \Delta_q(S_i) \text{ is adjacent to the new segment } s_i. \]

\[ \Rightarrow \text{top}(\Delta) = s_i, \text{ bot}(\Delta) = s_i, \text{ leftp}(\Delta) \in s_i, \text{ or rightp}(\Delta) \in s_i. \]

**Trick:** \( \mathcal{T}(S_i) \) (and thus \( \Delta \)) is uniquely determined by \( S_i \).

Consider \( S_i \subseteq S \) fixed.

\[ \Rightarrow \Delta \text{ does not depend on insertion order.} \]
Query Time (cont’d)

\( p_i = \text{prob. that the search path } \Pi_q \text{ of } q \text{ in } D \text{ contains a node that was created in iteration } i. \)

i.e., prob that \( \Delta \) changes when inserting \( s_i \).

**Aim:** bound \( p_i \).

**Tool:** *Backwards analysis!*

\( p_i = \text{prob that } \Delta \text{ changes when } s_i \text{ is removed} \)

Four cases:

\[
P(\text{top}(\Delta) = s_i) = \frac{1}{i} \quad \text{(since exactly one of } i \text{ segments is top}(\Delta)).
\]

\[\Rightarrow p_i \leq \frac{4}{i}\]

\[\Rightarrow E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i] \leq \sum_{i=1}^{n} 3 \cdot p_i
\]

\[= 12 \sum_{i=1}^{n} \frac{1}{i} \leq O(\log n)\]