Computational Geometry

Simple Range Searching

Lecture #11
Range-Counting Query

area affected by the construction of a new airport

Observation.

Query range depends on, e.g., dominant wind directions

⇒ non-orthogonal
Non-orthogonal range queries

Query range:

Problem. Given a set $P$ of $n$ points, preprocess $P$ such that half-space range-counting queries can be answered quickly.

Task Design a data structure for the 1-dim. case:

– Given a number $x$, return $|P \cap [x, \infty)|$.

– Consider $P$ static / dynamic!
The 1-Dimensional Case

**Task.** Design a data structure for the 1-dim. case!

**Solution.**
- use balanced binary search trees
- augment each node with the number of nodes in its subtree [see Cormen et al., *Introduction to Algorithms*, MIT press, 3rd ed., 2009]

**Lesson.** On each level, visit \( \leq 1 \) subtree recursively!
Generalizing to 2 Dimensions

Partition the input! Query... in a partition tree ... recursively!

Definition. $\Psi(S) = \{(S_1, t_1), (S_2, t_2), \ldots, (S_r, t_r)\}$ is a simplicial partition (of size $r$) for $S$ if

- $S$ is partitioned by $S_1, \ldots, S_r$ and
- for $1 \leq i \leq r$, $t_i$ is a triangle and $S_i \subset t_i$.

$\Psi(S)$ is fine if $|S_i| \leq 2\frac{|S|}{r}$ for every $1 \leq i \leq r$. 
**Generalizing to 2 Dimensions**

Partition the input! Query... in a *partition tree* ... recursively!

**Definition.** The *crossing number* of $\ell$ (w.r.t. $\Psi(S)$) is the number of triangles $t_1, \ldots, t_r$ crossed by $\ell$.

The *crossing number* of $\Psi(S)$ is the maximum crossing number over all possible lines.
Generalizing to 2 Dimensions

Partition the input! Query... in a partition tree... recursively!

**Theorem.** [Matoušek, DCG 1992] For any set $S$ of $n$ pts and any $1 \leq r \leq n$, a fine simplicial partition of size $r$ and crossing number $O(\sqrt{r})$ exists. For any $\varepsilon > 0$, such a partition can be built in $O(n^{1+\varepsilon})$ time.

**Lemma.** A partition tree for $S$ can be constructed in $O(n^{1+\varepsilon})$ time. The tree uses $O(n)$ storage.
Example for a Query

point set $S$

$h$: query range

partition tree for $S$

partition by triangles

= selected node

= visited node

recursively visited subtrees
**Query Algorithm**

`SelectInHalfplane` (half-plane $h$, partit. tree $T$ for pt set $S$)

$N \leftarrow \emptyset$  

// set of selected nodes

**if** $T = \{\mu\}$ **then**

**if** point stored at $\mu$ lies in $h$ **then**

$N \leftarrow \{\mu\}$

else

**foreach** child $\nu$ of the root of $T$ **do**

**if** $t(\nu) \subset h$ **then**

$N \leftarrow N \cup \{\nu\}$

else

**if** $t(\nu) \cap h \neq \emptyset$ **then**

$N \leftarrow N \cup SelectInHalfplane(h, T_\nu)$

return $N$  

// with $S \cap h = \bigcup_{\nu \in N} S(\nu)$
Query Algorithm

SelectInHalfplane(half-plane $h$, partit. tree $T$ for pt set $S$)

$N \leftarrow \emptyset$  // set of selected nodes

if $T = \{\mu\}$ then
  if point stored at $\mu$ lies in $h$ then
    $N \leftarrow \{\mu\}$
  else
    foreach child $\nu$ of the root of $T$ do
      if $t(\nu) \subseteq h$ then
        $N \leftarrow N \cup \{\nu\} + |S(\nu)|$
      else
        if $t(\nu) \cap h \neq \emptyset$ then
          $N \leftarrow N \cup \text{SelectInHalfplane}(h, T_\nu)$ + \text{Count}

return $N$  // with $|S \cap h| = \bigcup_{v \in N} S(v)$

Task:

Turn this into a range counting query algorithm!
Analysis of the Partition Tree

**Lemma.** For any \( \varepsilon > 0 \), there is a partition tree \( \mathcal{T} \) for \( S \) s.t.:
for a query half-plane \( h \),
\( \text{SELECTINHALFPLANE} \) selects in \( O(n^{1/2+\varepsilon}) \) time
a set \( N \) of \( O(n^{1/2+\varepsilon}) \) nodes of \( \mathcal{T} \)
with the property that \( h \cap S = \bigcup_{v \in N} S(v) \).

**Proof.** Let \( \varepsilon > 0 \). Let \( r = 2(\sqrt{2}c)^{1/\varepsilon} \).
\[ Q(n) \leq \begin{cases} 1 & \text{if } n = 1, \\ r + \sum_{v \in C(h)} Q(|S(v)|) & \text{if } n > 1. \end{cases} \]

\( C(h) \) : all children \( v \) of the root s.t. \( h \) crosses \( t(v) \)

**Theorem.** For any set \( S \) of \( n \) pts and any \( 1 \leq r \leq n \), a **fine** simplicial partition of size \( r \) and crossing number \( c\sqrt{r} \) exists. For any \( \varepsilon > 0 \), such a partition can be built in \( O(n^{1+\varepsilon}) \) time.

[Matoušek, DCG 1992]
Analysis of the Partition Tree

**Lemma.** A partition tree for $S$ can be constructed in $O(n^{1+\varepsilon})$ time. The tree uses $O(n)$ storage.

**Lemma.** For any $\varepsilon > 0$, there is a partition tree $\mathcal{T}$ for $S$ s.t.: for a query half-plane $h$, SelectInHalfplane selects in $O(n^{1/2+\varepsilon})$ time a set $N$ of $O(n^{1/2+\varepsilon})$ nodes of $\mathcal{T}$ with the property that $h \cap S = \bigcup_{\nu \in N} S(\nu)$.

**Corollary.** Half-plane range counting queries can be answered in $O(n^{1/2+\varepsilon})$ time using $O(n)$ space and $O(n^{1+\varepsilon})$ prep.
Any ideas? Just use SelectInHalfplane!

Theorem. Given a set $S$ of $n$ pts in the plane, for any $\varepsilon > 0$, a triangular range-counting query can be answered in $O(n^{1/2+\varepsilon})$ time using a partition tree. The tree can be built in $O(n^{1+\varepsilon})$ time and uses $O(n)$ space. The points inside the query range can be reported in $O(k)$ additional time, where $k$ is the number of reported pts.

Can we do better?

Use cutting trees! (Chapter 16.3) Query time $O(\log^3 n)$, prep. & storage $O(n^{2+\varepsilon})$. 
Multi-Level Partition Trees

Idea. Store with each internal node not just a number, but another data structure!

Task. Design a fast data structure for line segments that counts all segments intersecting a query line $\ell$. 

$p_{\text{left}}(s')$ $p_{\text{right}}(s')$
Query Algorithm

\[
N \leftarrow \emptyset
\]

\[
\text{if } T = \{\mu\} \text{ then}
\]

\[
\text{if segment stored in } \mu \text{ intersects } \ell \text{ then } N \leftarrow \{\mu\}
\]

\[
\text{else}
\]

\[
\text{foreach child } \nu \text{ of } \mathcal{T}'s \text{ root do}
\]

\[
\text{if } t(\nu) \subseteq \ell^+ \text{ then}
\]

\[
N \leftarrow N \cup \text{SelectInHalfplane}(\ell^-, \mathcal{T}_{assoc}^\nu)
\]

\[
\text{else}
\]

\[
\text{if } t(\nu) \cap \ell \neq \emptyset \text{ then}
\]

\[
N \leftarrow N \cup \text{SelectIntSegments}(\ell, \mathcal{T}_\nu)
\]

\[
\text{return } N
\]

For \( S' \subseteq S \), let

\[
P_{\text{right}}(S') = \{p_{\text{right}}(s) \mid s \in S'\}
\]

-- first-level tree stores \( P_{\text{right}}(S) \)

-- second-level trees store subsets of \( P_{\text{left}}(S) \)

stores \( P_{\text{left}}(S_{\text{seg}}(\nu)) \), where

\[
S_{\text{seg}}(\nu) = \{s \mid p_{\text{right}}(s) \in S(\nu)\}
\]

For \( S' \subseteq S \), let

\[
P_{\text{left}}(S') = \{p_{\text{left}}(s) \mid s \in S'\}
\]

\[
\bigcup_{\nu \in N} S(\nu) = \{s \in S \mid p_{\text{right}}(s) \text{ above } \ell \text{ and } p_{\text{left}}(s) \text{ below } \ell\}
\]
Results

Lemma. A 2-level partition tree for line-intersection queries among a set of $n$ segments uses $O(n \log n)$ storage.

Lemma. Let $S$ be a set of $n$ segments in the plane. For any $\varepsilon > 0$, there is a 2-level partition tree $\mathcal{T}$ for $S$ s.t.

- given a query line $\ell$, we can select $O(n^{1/2+\varepsilon})$ nodes from $\mathcal{T}$ whose canonical subsets represent the segments intersected by $\ell$.

- The selection takes $O(n^{1/2+\varepsilon})$ time.

Corollary. Let $S$ be a set of $n$ segments in the plane. We can count the number of segments in $S$ intersected by a query line in $O(n^{1/2+\varepsilon})$ time using $O(n \log n)$ space and $O(n^{1+\varepsilon})$ prep.
Results

Lemma. A 2-level partition tree for line-intersection queries among a set of $n$ segments uses $O(n \log n)$ storage.

Lemma. Let $S$ be a set of $n$ segments in the plane. For any $\varepsilon > 0$, there is a 2-level partition tree $T$ for $S$ s.t.

- given a query line $\ell$, we can select $O(n^{1/2+\varepsilon})$ nodes from $T$ whose canonical subsets represent the segments intersected by $\ell$.
- The selection takes $O(n^{1/2+\varepsilon})$ time.

Corollary. Let $S$ be a set of $n$ segments in the plane. We can count the number of segments in $S$ intersected by a query line in $O(n^{1/2+\delta\varepsilon})$ time using $O(n \log^{\delta-1} n)$ space and $O(n^{1+\delta\varepsilon})$ prep.