Computation Geometry – Exercise
Range Searching II

Guido Brückner
20.07.2018
Object types in range queries

Setting so far:
- Input: set of points \( P \)
  (here \( P \subset \mathbb{R}^2 \))
- Output: all points in \( P \cap [x, x'] \times [y, y'] \)
- Data structures: \( kd \)-trees or range trees
Object types in range queries

Setting so far:
• Input: set of points \( P \) (here \( P \subset \mathbb{R}^2 \))
• Output: all points in \( P \cap [x, x'] \times [y, y'] \)
• Data structures: \( kd \)-trees or range trees

Further variant
• Input: set of line segments \( S \) (here in \( \mathbb{R}^2 \))
• Output: all segments in \( S \cap [x, x'] \times [y, y'] \)
• Data structures: ?
Axis-parallel line segments

special case (e.g., in VLSI design): all line segments are axis-parallel
Axis-parallel line segments

special case (e.g., in VLSI design): all line segments are axis-parallel

Problem:

Given $n$ vertical and horizontal line segments and an axis-parallel rectangle $R = [x, x'] \times [y, y']$, find all line segments that intersect $R$. 
Axis-parallel line segments

special case (e.g., in VLSI design): all line segments are axis-parallel

Problem:
Given $n$ vertical and horizontal line segments and an axis-parallel rectangle $R = [x, x'] \times [y, y']$, find all line segments that intersect $R$.

How to approach this case?
Axis-parallel line segments

special case (e.g., in VLSI design): all line segments are axis-parallel

Problem:

Given $n$ vertical and horizontal line segments and an axis-parallel rectangle $R = [x, x'] \times [y, y']$, find all line segments that intersect $R$.

Case 1: $\geq 1$ endpoint in $R$
$\rightarrow$ use range tree

Case 2: both endpoints $\not\in R$
$\rightarrow$ intersect left or top edge of $R$
Case 2 in detail

Problem:
Given a set $H$ of $n$ horizontal line segments and a vertical query segment $s$, find all line segments in $H$ that intersect $s$. (Vertical segments and a horizontal query are analogous.)
Case 2 in detail

Problem:
Given a set $H$ of $n$ horizontal line segments and a vertical query segment $s$, find all line segments in $H$ that intersect $s$. (Vertical segments and a horizontal query are analogous.)

One level simpler: vertical line $s := (x = q_x)$
Given $n$ intervals $I = \{[x_1, x'_1], [x_2, x'_2], \ldots, [x_n, x'_n]\}$ and a point $q_x$, find all intervals that contain $q_x$. 
Case 2 in detail

Problem:
Given a set $H$ of $n$ horizontal line segments and a vertical query segment $s$, find all line segments in $H$ that intersect $s$. (Vertical segments and a horizontal query are analogous.)

One level simpler: vertical line $s := (x = q_x)$

Given $n$ intervals $I = \{[x_1, x'_1], [x_2, x'_2], \ldots, [x_n, x'_n]\}$ and a point $q_x$, find all intervals that contain $q_x$.

What do we need for an appropriate data structure?
Interval Trees

Construction of an interval tree \( \mathcal{T} \)

- if \( I = \emptyset \) then \( \mathcal{T} \) is a leaf
- else let \( x_{\text{mid}} \) be the median of the endpoints of \( I \) and define

\[
\begin{align*}
I_{\text{left}} &= \{ [x_j, x'_j] \mid x'_j < x_{\text{mid}} \} \\
I_{\text{mid}} &= \{ [x_j, x'_j] \mid x_j \leq x_{\text{mid}} \leq x'_j \} \\
I_{\text{right}} &= \{ [x_j, x'_j] \mid x_{\text{mid}} < x_j \}
\end{align*}
\]

\( \mathcal{T} \) consists of a node \( v \) for \( x_{\text{mid}} \) and

- lists \( \mathcal{L}(v) \) and \( \mathcal{R}(v) \) for \( I_{\text{mid}} \) sorted by left and right interval endpoints, respectively
- left child of \( v \) is an interval tree for \( I_{\text{left}} \)
- right child of \( v \) is an interval tree for \( I_{\text{right}} \)

\[
\mathcal{L} = s_3, s_4, s_5 \\
\mathcal{R} = s_5, s_3, s_4
\]

\[
\mathcal{L} = s_1, s_2 \\
\mathcal{R} = s_1, s_2
\]

\[
\mathcal{L} = s_6, s_7 \\
\mathcal{R} = s_7, s_6
\]
From lines to line segments

How can we adapt the idea of an interval tree for query segments $q_x \times [q_y, q'_y]$ instead of a query line $x = q_x$?
From lines to line segments

How can we adapt the idea of an interval tree for query segments $q_x \times [q_y, q'_y]$ instead of a query line $x = q_x$?
From lines to line segments

How can we adapt the idea of an interval tree for query segments $q_x \times [q_y, q'_y]$ instead of a query line $x = q_x$?

The correct line segments in $I_{\text{mid}}$ can easily be found using a range tree instead of simple lists.
From lines to line segments

How can we adapt the idea of an interval tree for query segments \( qx \times [q_y, q'_y] \) instead of a query line \( x = qx \)?

The correct line segments in \( I_{mid} \) can easily be found using a range tree instead of simple lists.

**Theorem 1:** Let \( S \) be a set of horizontal (axis-parallel) line segments in the plane. All \( k \) line segments that intersect a vertical query segment (an axis-parallel rectangle \( R \)) can be found in \( O(\log^2(n) + k) \) time. The data structure requires \( O(n \log n) \) space and construction time.
Arbitrary line segments

Map data often contain arbitrarily oriented line segments.

Problem:
Given \( n \) disjoint line segments and an axis-parallel rectangle \( R = [x, x'] \times [y, y'] \), find all line segments that intersect \( R \).

Exercise 5: How to use interval trees?
Solution

Use Bounding Box of Segments

1. Interval trees on segments of bounding-boxes.

2. If segments of bounding-box intersect query range:
   
   Check whether contained segment intersects query range.
Solution

Use Bounding Box of Segments
1. Interval trees on segments of bounding-boxes.

2. If segments of bounding-box intersect query range:
   Check whether contained segment intersects query range.

Correct, because:
Solution

Use Bounding Box of Segments

1. Interval trees on segments of bounding-boxes.

2. If segments of bounding-box intersect query range:
   Check whether contained segment intersects query range.

Correct, because:

If a segment intersects the query range $R$, then the corresponding bounding box intersects $R$. 
Solution

Use Bounding Box of Segments
1. Interval trees on segments of bounding-boxes.
2. If segments of bounding-box intersect query range:
   - Check whether contained segment intersects query range.

Correct, because:
- If a segment intersects the query range $R$, then the corresponding bounding box intersects $R$.

Problem:
Solution

Use Bounding Box of Segments

1. Interval trees on segments of bounding-boxes.

2. If segments of bounding-box intersect query range:
   
   Correct, because:
   
   If a segment intersects the query range $R$, then the corresponding bounding box intersects $R$.

   Problem: More segments may be considered than necessary.

   because it is not true that

   If the bounding box intersects the query range, then the contained segment intersects the query range.
Arbitrary line segments

Map data often contain arbitrarily oriented line segments.

Problem:
Given $n$ disjoint line segments and an axis-parallel rectangle $R = [x, x'] \times [y, y']$, find all line segments that intersect $R$.

How to proceed?
Arbitrary line segments

Map data often contain arbitrarily oriented line segments.

Problem:
Given \( n \) disjoint line segments and an axis-parallel rectangle \( R = [x, x'] \times [y, y'] \), find all line segments that intersect \( R \).

How to proceed?

Case 1: \( \geq 1 \) endpoint in \( R \) \( \rightarrow \) use range tree
Case 2: both endpoints \( \notin R \) \( \rightarrow \) intersect at least one edge of \( R \)
Decomposition into elementary intervals

Interval trees don’t really help here

\[ [-\infty, q_x] \times [q_y, q_y'] \]
Decomposition into elementary intervals

Interval trees don’t really help here

\[ [−∞, q_x] \times [q_y, q'_y] \]

Identical 1d base problem:
Given \( n \) intervals \( I = \{[x_1, x'_1], [x_2, x'_2], \ldots, [x_n, x'_n]\} \) and a point \( q_x \), find all intervals that contain \( q_x \).

• sort all \( x_i \) and \( x'_i \) in list \( p_1, \ldots, p_{2n} \)
• create sorted elementary intervals
  \( (−∞, p_1), [p_1, p_1], (p_1, p_2), [p_2, p_2], \ldots, [p_{2n}, p_{2n}], (p_{2n}, ∞) \)
**Segment trees**

Idea for data structure:
- create binary search tree with elementary intervals in leaves
- for all points $q_x$ in the same elementary interval the answer is the same
- leaf $\mu$ for elementary interval $e(\mu)$ stores interval set $I(\mu)$
- query requires $O(\log n + k)$ time
Segment trees

Idea for data structure:
• create binary search tree with elementary intervals in leaves
• for all points $q_x$ in the same elementary interval the answer is the same
• leaf $\mu$ for elementary interval $e(\mu)$ stores interval set $I(\mu)$
• query requires $O(\log n + k)$ time

Store intervals as high up in the tree as possible
• node $v$ represents interval $e(v) = e(lc(v)) \cup e(rc(v))$
• input interval $s_i \in I(v) \iff e(v) \subseteq s_i$ and $e(parent(v)) \not\subseteq s_i$
Segment trees

Idea for data structure:
• create binary search tree with elementary intervals in leaves
• for all points \( q_x \) in the same elementary interval the answer is the same
• leaf \( \mu \) for elementary interval \( e(\mu) \) stores interval set \( I(\mu) \)
• query requires \( O(\log n + k) \) time

Store intervals as high up in the tree as possible
• node \( v \) represents interval \( e(v) = e(lc(v)) \cup e(rc(v)) \)
• input interval \( s_i \in I(v) \iff e(v) \subseteq s_i \) and \( e(parent(v)) \nsubseteq s_i \)
Back to arbitrary line segments

- create segment tree for the $x$ intervals of the line segments
- each node $v$ corresponds to a vertical strip $e(v) \times \mathbb{R}$
- line segment $s$ is in $I(v)$ iff $s$ crosses the strip of $v$ but not the strip of $\text{parent}(v)$
- at each node $v$ on the search path for the vertical segment $s' = q_x \times [q_y, q'_y]$ all segments in $I(v)$ cover the $x$-coordinate $q_x$
Back to arbitrary line segments

- create segment tree for the $x$ intervals of the line segments
- each node $v$ corresponds to a vertical strip $e(v) \times \mathbb{R}$
- line segment $s$ is in $I(v)$ iff $s$ crosses the strip of $v$ but not the strip of $\text{parent}(v)$
- at each node $v$ on the search path for the vertical segment $s' = q_x \times [q_y, q'_y]$ all segments in $I(v)$ cover the $x$-coordinate $q_x$
- find segments in the strip that cross $s'$ using a vertically sorted auxiliary binary search tree
Summary

Theorem 2: Let $S$ be a set of interior-disjoint line segments in the plane. All $k$ segments that intersect a vertical query segment (an axis-parallel query rectangle $R$) can be found in time $O(k + \log^2 n)$. The corresponding data structure uses $O(n \log n)$ space and $O(n \log^2 n)$ construction time.
Summary

Theorem 2: Let $S$ be a set of interior-disjoint line segments in the plane. All $k$ segments that intersect a vertical query segment (an axis-parallel query rectangle $R$) can be found in time $O(k + \log^2 n)$. The corresponding data structure uses $O(n \log n)$ space and $O(n \log^2 n)$ construction time.

Remark:
The construction time for the data structure can be improved to $O(n \log n)$. 
Summary

**Theorem 2:** Let $S$ be a set of interior-disjoint line segments in the plane. All $k$ segments that intersect a vertical query segment (an axis-parallel query rectangle $R$) can be found in time $O(k + \log^2 n)$. The corresponding data structure uses $O(n \log n)$ space and $O(n \log^2 n)$ construction time.

**Remark:**
The construction time for the data structure can be improved to $O(n \log n)$.

**Problem:** Construction of auxiliary tree.
Solution

Let $s_1, s_2$ be two segments.

$s_1$ lies below $s_2$ ($s_1 \prec s_2$), if there is a point $p_1 \in s_1$ and $p_2 \in s_2$ with $x(p_1) = x(p_2)$ and $y(p_1) < y(p_2)$.

1. Show that relation $\prec$ on $S$ is acyclic.

   There exists a topological ordering.

2. Compute topological ordering $S$

3. Use topological ordering to construct help trees.
Solution

Let \( s_1, s_2 \) be two segments.

\( s_1 \) lies below \( s_2 \) (\( s_1 \prec s_2 \)), if there is a point \( p_1 \in s_1 \) and \( p_2 \in s_2 \) with \( x(p_1) = x(p_2) \) and \( y(p_1) < y(p_2) \).

1. Show that relation \( \prec \) on \( S \) is acyclic.

   ➔ There exists a topological ordering.

2. Compute topological ordering \( S \)

3. Use topological ordering to construct help trees.
Computation of Topological Ordering

Verical sweep-line from left to right to obtain ordering $T$:

**Events:** Endpoints of segments.

**Sweep-Line-State:** Segments that intersect sweep-line (binary tree $S$ representation)

**Handling event $p$:**

$p$ ist left end point of segment $s_i$: insert $s_i$ into $S$.

$p$ is right end point of segment $s_i$: $s_i$ is removed from $S$.

Insert $s_i$ into $T$ correspondingly to its neighbors in $S$. 
Computation of Topological Ordering

Verical sweep-line from left to right to obtain ordering $T$:

**Events:** Endpoints of segments.

**Sweep-Line-State:** Segments that intersect sweep-line (binary tree $S$ representation)

**Handling event $p$:**

$p$ is left end point of segment $s_i$: insert $s_i$ into $S$.

$S$:

- $s_1$
- $s_2$
- $s_3$
- $s_4$
- $s_5$

$T$:

- $s_1$
- $s_3$
- $s_5$
- $s_2$
- $s_4$
Computation of Topological Ordering

Vertical sweep-line from left to right to obtain ordering $T$:

**Events:** Endpoints of segments.

**Sweep-Line-State:** Segments that intersect sweep-line (binary tree $S$ representation)

**Handling event $p$:**

$p$ ist left end point of segment $s_i$: insert $s_i$ into $S$.

Insert $s_i$ into $T$ correspondingly to its neighbors in $S$.

$p$ is right end point of segment $s_i$: $s_i$ is removed from $S$.
Computation of Topological Ordering

![Diagram](image)

Vertical sweep-line from left to right to obtain ordering $T$:

**Events:** Endpoints of segments.

**Sweep-Line-State:** Segments that intersect sweep-line (binary tree $S$ representation)

**Handling event $p$:**

- $p$ is left end point of segment $s_i$: insert $s_i$ into $S$.
- Insert $s_i$ into $T$ correspondingly to its neighbors in $S$.
- $p$ is right end point of segment $s_i$: $s_i$ is removed from $S$.
Computation of Topological Ordering

Events: Endpoints of segments.

Sweep-Line-State: Segments that intersect sweep-line (binary tree \( S \) representation)

Handling event \( p \):

\( p \) ist left end point of segment \( s_i \): insert \( s_i \) into \( S \).

Insert \( s_i \) into \( T \) correspondingly to its neighbors in \( S \).

\( p \) is right end point of segment \( s_i \): \( s_i \) is removed from \( S \).
Computation of Topological Ordering

Verical sweep-line from left to right to obtain ordering $T$:

**Events:** Endpoints of segments.

**Sweep-Line-State:** Segments that intersect sweep-line (binary tree $S$ representation)

**Handling event $p$:**

$p$ ist left end point of segment $s_i$: insert $s_i$ into $S$.

Insert $s_i$ into $T$ correspondingly to its neighbors in $S$.

$p$ is right end point of segment $s_i$: $s_i$ is removed from $S$.
Construction of Trees

Apply topological ordering.
Construction of Trees

Apply topological ordering.

In each strip the topological ordering corresponds with the vertical ordering.

- Insert segments into \( I(v) \) w.r.t. topological ordering.
- Construct binary tree based on \( I(v) \) in \( |I(v)| \) time.
- \( O(n) \) time in total.
Exercise 3

given: Set $I$ of $n$ intervals

find: In how many intervals is a point $p \in \mathbb{R}$ contained? Datastructure!

Datastructure is based on interval trees:
Interval Trees

Construction of an interval tree $T$

- if $I = \emptyset$ then $T$ is a leaf
- else let $x_{\text{mid}}$ be the median of the endpoints of $I$ and define

\[
\begin{align*}
I_{\text{left}} &= \{ [x_j, x'_j] \mid x'_j < x_{\text{mid}} \} \\
I_{\text{mid}} &= \{ [x_j, x'_j] \mid x_j \leq x_{\text{mid}} \leq x'_j \} \\
I_{\text{right}} &= \{ [x_j, x'_j] \mid x_{\text{mid}} < x_j \}
\end{align*}
\]

$T$ consists of a node $v$ for $x_{\text{mid}}$ and

- lists $L(v)$ and $R(v)$ for $I_{\text{mid}}$ sorted by left and right interval endpoints, respectively
- left child of $v$ is an interval tree for $I_{\text{left}}$
- right child of $v$ is an interval tree for $I_{\text{right}}$
Exercise 3

given: Set $I$ of $n$ intervals
find: In how many intervals is a point $p \in \mathbb{R}$ contained? Datastructure!

Data structure is based on interval trees:

$\text{QIT}(v, q_x)$

\[
\text{if } v \text{ is not Blatt then} \quad \text{if } q_x < x_{\text{mid}}(v) \text{ then} \quad \text{return } \text{QIT}(\text{lc}(v), q_x) + \text{Number of intervals in } \mathcal{L} \text{ that contain } q_x \\
\text{else} \quad \text{return } \text{QIT}(\text{rc}(v), q_x) + \text{Number of intervals in } \mathcal{R} \text{ that contain } q_x
\]

\text{return 1}
Exercise 3

given: Set \( I \) of \( n \) intervals

find: In how many intervals is a point \( p \in \mathbb{R} \) contained? Datastructure!

Data structure is based on interval trees:

\[
\text{QIT}(v, q_x)
\]

if \( v \) is not Blatt then
  
  if \( q_x < x_{\text{mid}}(v) \) then
    return \( \text{QIT}(\text{lc}(v), q_x) + \text{Number of intervals in } \mathcal{L} \text{ that contain } q_x \)
  else
    return \( \text{QIT}(\text{rc}(v), q_x) + \text{Number of intervals in } \mathcal{R} \text{ that contain } q_x \)

return 1

Running Time: \( O(\log^2 n) \)
Exercise 3

given: Set \( I \) of \( n \) intervals

find: In how many intervals is a point \( p \in \mathbb{R} \) contained? Datastructure!

Data structure based on segment trees:
Exercise 3

given: Set $I$ of $n$ intervals
find: In how many intervals is a point $p \in \mathbb{R}$ contained? Datastructure!

Data structure based on segment trees:

\[
\text{QuerySegmentTree}(v, q_x) \\
\text{if } v \text{ is not leaf then} \\
\quad \text{if } q_x \in e(lc(v)) \text{ then} \\
\quad \quad \text{QuerySegmentTree}(lc(v), q_x) + |I(v)| \\
\quad \text{else} \\
\quad \quad \text{QuerySegmentTree}(rc(v), q_x) + |I(v)| \\
\text{return 1}
\]

Store $|I(v)|$ instead of $I(v)$

$O(\log n)$ time and $O(n)$ storage.
Exercise 3

given: Set $I$ of $n$ intervals

find: In how many intervals is a point $p \in \mathbb{R}$ contained? Datastructure!

Data structure based on binary tree.
Exercise 3

given: Set $I$ of $n$ intervals

find: In how many intervals is a point $p \in \mathbb{R}$ contained? Datastructure!

Data structure based on binary tree.

1. Split intervals into elementary intervals.
2. Store for each elem. interval, in how many intervals it is contained.
3. Construct binary tree based on borders of elem. intervals.

$\text{Time } O(\log n), \text{ Storage } O(n)$
Exercise 4

**Given**: Set $\mathcal{R}$ of axis-aligned rectangles.

**Find**: Algorithm that computes $\max_{p \in \mathcal{R}} w_{\mathcal{R}}(p)$ in $O(n \log n)$ time.

For $p \in \mathbb{R}$, $w_{\mathcal{R}}(p)$ is the number of rectangles in $\mathcal{R}$ that contain $p$. 
Exercise 4

Given: Set $\mathcal{R}$ of axis-aligned rectangles.
Find: Algorithm that computes $\max_{p \in \mathcal{R}} w_\mathcal{R}(p)$ in $O(n \log n)$ time.

For $p \in \mathbb{R}$, $w_\mathcal{R}(p)$ is the number of rectangles in $\mathcal{R}$ that contain $p$.

Sweep-Line: from left to right
SL-State: segment tree $T$ that stores vertical edges as intervals.
Events: vertical edges of rectangles.

1. Determine the number or intervals in $T$ intersecting $[y(p), y(q)]$.
2. Insert $[y(p), y(q)]$ into $T$.
3. Update $\max w_\mathcal{R}(p)$
Exercise 4

Given: Set $\mathcal{R}$ of axis-aligned rectangles.

Find: Algorithm that computes $\max_{p \in \mathcal{R}} w_{\mathcal{R}}(p)$ in $O(n \log n)$ time.

For $p \in \mathbb{R}$, $w_{\mathcal{R}}(p)$ is the number of rectangles in $\mathcal{R}$ that contain $p$.

Sweep-Line: from left to right

SL-State: segment tree $T$ that stores vertical edges as intervals.

Events: vertical edges of rectangles.

- right vert. edge $pq$: delete interval $[y(p), y(q)]$. 