Lecture

Range Searching II: Windowing Queries

Tamara Mchedlidze · Darren Strash
23.11.2015
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

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 $R$
→ use range tree

Case 2: both endpoints $\not\in R$
→ 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.)

**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 $T$

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

$$I_{\text{left}} = \{ [x_j, x'_j] \mid x'_j < x_{\text{mid}} \}$$
$$I_{\text{mid}} = \{ [x_j, x'_j] \mid x_j \leq x_{\text{mid}} \leq x'_j \}$$
$$I_{\text{right}} = \{ [x_j, x'_j] \mid x_{\text{mid}} < x_j \}$$

$T$ 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 $I_{\text{right}}$
Properties of interval trees

**Lemma 1:** An interval tree for $n$ intervals needs $O(n)$ space and has depth $O(\log n)$. It can be constructed in time $O(n \log n)$.

```plaintext
QueryIntervalTree(v, qx)
    if v no leaf then
        if qx < x_{mid}(v) then
            search in $L$ from left to right for intervals containing $q_x$
            QueryIntervalTree(lc(v), qx)
        else
            search in $R$ from right to left for intervals containing $q_x$
            QueryIntervalTree(rc(v), qx)
```

**Lemma 2:** Using an interval tree we can find all $k$ intervals containing a query point $q_x$ in $O(\log n + k)$ time.
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.

**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 $R$.

How to proceed?

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

Case 2: both endpoints $\not\in R \rightarrow$ intersect at least one edge of $R$
Decomposition into elementary intervals

Interval trees don’t really help here

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
  $(-\infty, p_1), [p_1, p_1], (p_1, p_2), [p_2, p_2], \ldots, [p_{2n}, p_{2n}], (p_{2n}, \infty)$
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

**Problem:** Storage space is worst-case quadratic

→ 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 $e(parent(v)) \not\subseteq s_i$
Properties of segment trees

**Lemma 3:** A segment tree for \( n \) intervals requires \( O(n \log n) \) space and can be constructed in \( O(n \log n) \) time.

Sketch of proof:

\[
\text{InsertSegmentTree}(v, [x, x'])
\]

\[
\begin{array}{l}
\text{if } e(v) \subseteq [x, x'] \text{ then} \\
\quad \text{store } [x, x'] \text{ in } I(v) \\
\text{else} \\
\quad \text{if } e(lc(v)) \cap [x, x'] \neq \emptyset \text{ then} \\
\quad \quad \text{InsertSegmentTree}(lc(v), [x, x']) \\
\quad \text{if } e(rc(v)) \cap [x, x'] \neq \emptyset \text{ then} \\
\quad \quad \text{InsertSegmentTree}(rc(v), [x, x'])
\end{array}
\]
Queries in segment trees

QuerySegmentTree($v, q_x$)

return all intervals in $I(v)$

if $v$ no leaf then
  if $q_x \in e(lc(v))$ then
    QuerySegmentTree($lc(v), q_x$)
  else
    QuerySegmentTree($rc(v), q_x$)

Lemma 4: All $k$ intervals that contain a query point $q_x$ can be computed in $O(\log n + k)$ time using a segment tree.

Lemma 4 yields the same result as interval trees. What is different?

→ all intervals stored in a positive node $v$ contain $q_x$ – in an interval tree one would have to continue searching
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 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.

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

Space requirement of interval trees

We have used range trees with $O(n \log n)$ space as auxiliary data structure in the interval trees. Using modified heaps this can be reduced to $O(n)$, see Chapter 10.2 in [BCKO’08].

How can you efficiently count the intersected segments?

Segment and interval trees support efficient counting queries (independent of $k$) with minor modifications → see exercises.

What to do for non-rectangular query regions?

By triangulating the query polygon, the problem can be reduced to triangular queries. Suitable data structures can be found, e.g., in chapter 16 of [BCKO’08].