Segment Trees

[Bentley]

A *segment tree* is a data structure for storing a set of intervals

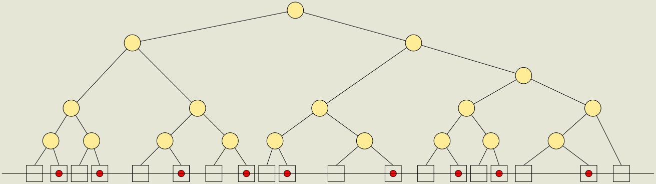
$$I = \{[x_1, x'_1], [x_2, x'_2], \ldots, [x_n, x'_n]\}$$

and can be used for solving problems e.g. concerning line segments.

Let $p_1, \ldots, p_m$, $m \leq 2n$, be the ordered list of distinct endpoints of the intervals in I. The ordered sequence of endpoints $p_1, \ldots, p_m$ partitions the real line into a set of *atomic intervals*

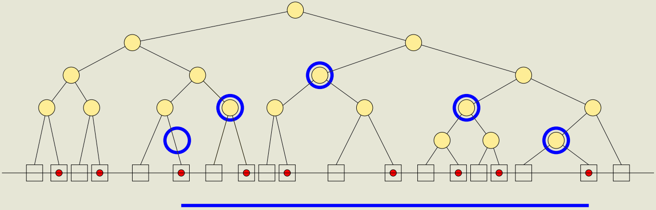
$$(-\infty, p_1), [p_1, p_1], (p_1, p_2), [p_2, p_2], \ldots, (p_{n-1}, p_n), [p_n, p_n], (p_n, \infty)$$

A segment tree is a balanced tree where each node corresponds to an interval. The leaves correspond to the atomic intervals according to left to right order. An internal node *u* corresponds to the union of the intervals corresponding to the leaves of the subtree rooted at *u*.



Let $int(v)$ denote the interval corresponding to node v. With each node v we store a set $I(v) \subseteq I$: Interval $[x, x']$ is stored in $I(v)$ if and only if

$$int(v) \subseteq [x, x'] \quad \text{and} \quad int(parent(v)) \not\subseteq [x, x']$$

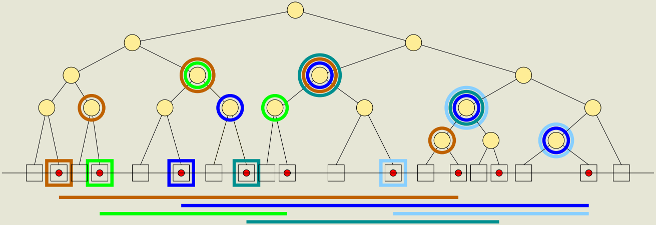


Lemma: A segment tree on n intervals uses $O(n \log n)$ storage.

An interval is stored with at most two nodes at the same depth of the tree.

```
InsertSegmentTree(v, [x, x'])
if int(v) ⊆ [x, x']
    then add [x, x'] to I(v)
    else  if int(lc(v)) ∩ [x, x'] ≠ ∅
             then InsertSegmentTree(lc(v), [x, x'])
          if int(rc(v)) ∩ [x, x'] ≠ ∅
             then InsertSegmentTree(rc(v), [x, x'])
```

Lemma: A segment tree for a set of n intervals can be constructed in $O(n \log n)$ time.



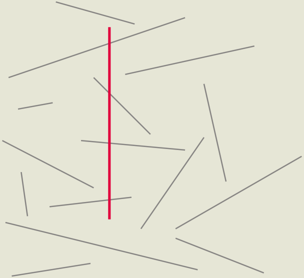
```
QuerySegmentTree(v, q_x)
Report all the intervals in I(v)
if v is not a leaf
   then if q_x ∈ int(lc(v))
           then QuerySegmentTree(lc(v), q_x)
           else QuerySegmentTree(rc(v), q_x)
```

Lemma: Using a segment tree, we can report all k intervals that contain a query point q_x, in time $O(k + \log n)$.

Vertical stabbing queries in a set of disjoint line segments

Let S be a set of pairwise disjoint line segments in the plane. We want to maintain S in a data structure that allows us to quickly find the segments intersected by a vertical query segment

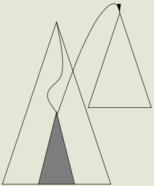
$$q_x \times [q_y, q'_y]$$

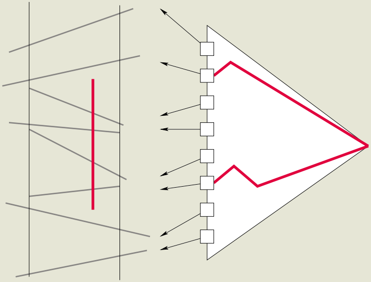


Store S in a segment tree based on the x-intervals of the segments.

For a node v, let $S(v)$ be the set of segments corresponding to the intervals in $I(v)$.

Store the segments in $S(v)$ in an associated balanced binary search tree based on the order of the elements in the slab $int(v) \times (-\infty, \infty)$.





If segments are horizontal, we get a segment-range tree.

Lemma: Let S be a set of n disjoint segments in the plane. S can be stored in a data structure such that the segments in S intersected by a vertical query segment can be reported in time $O(k + (\log n)^2)$, where k is the number of reported segments. The data structure uses $O(n \log n)$ storage space and can be built in $O(n(\log n)^2)$ time.

Construction time can be improved to $O(n \log n)$.

Semi-static Segment Trees

Let X be a set of N real numbers. We construct a balanced binary search tree on the atomic intervals defined by these numbers.

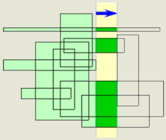
Now we can insert and delete intervals with endpoints in X. Insertion takes time $O(\log N)$. Deletion of an interval s takes time $O(\log N)$ as well, if we know the positions of s in all sets $I(v)$ where s is stored.

Klee's Measure Problem

Given a collection of intervals, what is the length of their union?

Given a collection of axis-aligned rectangles, what is the area of their union?

Use plane-sweep and maintain the intersection of the rectangles and the sweep line in a (semi-static) segment tree. [Bentley]



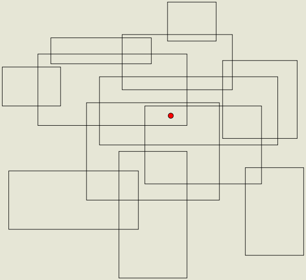
© Jeff Erickson

Theorem: The area of the union of n axis-aligned rectangles in the plane can be computed in $O(n \log n)$ time.

http://granmapa.cs.uiuc.edu/~jeffe/open/klee.html

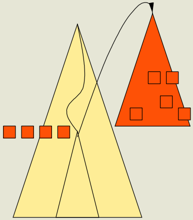
Higher-dimensional Segment Trees

Analogously to range trees, segment trees can be extended to higher dimensions, for example, in order to solve point enclosure problems for axis-aligned rectangular boxes. Such point enclosure problems are also called "inverse range queries".



A d-dimensional segment tree for axis-aligned rectangular boxes in $\mathbb{R}^d$ is an augmented one-dimensional segment tree for the atomic intervals according to the first dimension.

The secondary data structure associated with a node v is a $(d-1)$-dimensional segment tree according to the remaining coordinates for the boxes corresponding to the intervals stored at v. More precisely, it is a $(d-1)$-dimensional segment tree for the boxes formed by the remaining $(d-1)$ coordinates.



Theorem: A d-dimensional segment tree for a set of n axis-aligned rectangular boxes in $\mathbb{R}^d$ can be built in $O(n(\log n)^d)$ time and takes $O(n(\log n)^d)$ space.

Theorem: Using a d-dimensional segment tree, point enclosure queries for a set of n axis-aligned rectangular boxes in $\mathbb{R}^d$ can be answered in time $O(k + (\log n)^d)$ time, where k is the number of reported boxes.