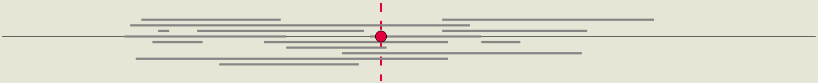
5

Stabbing Queries for Intervals

Given a set I of intervals

$$I = \{[x_1, x_1'], [x_2, x_2'], \ldots, [x_n, x_n']\}$$

report all elements in I intersected by a given query point.



Segment Tree

[J. L. Bentley; *Solutions to Klee's rectangle problem*, Technical Report, Carnegie-Mellon University, Pittsburgh, 1977]
[Section 10.3 in de Berg, Cheong, van Kreveld, Overmars; *Computational Geometry: Algorithms and Applications*, 3rd edition, 2008]
[Section 2.2 in de Langetepe Zachmann; *Geometric Data Structures for Computer Graphics*, 2006]

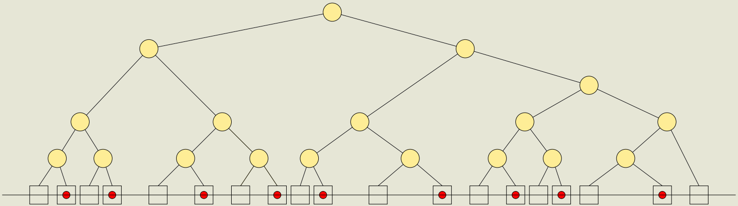
A *segment tree* is a static data structure for storing a set of intervals

$$I = \{[x_1, x_1'], [x_2, x_2'], \ldots, [x_n, x_n']\}$$

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*:

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

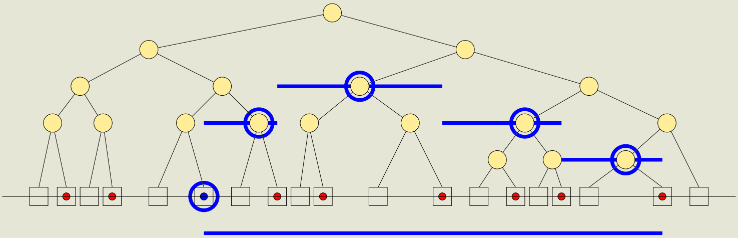


A segment tree is a leaf-oriented balanced binary tree on the atomic intervals according to left to right order.

An internal node v corresponds to the interval which is the union of the atomic intervals of the leaves of the subtree rooted at v.
Let $int(v)$ denote this interval.

With each node v we store a set $I(v) \subseteq I$:
Interval $[x, x']$ is stored in $I(v)$ if and only if

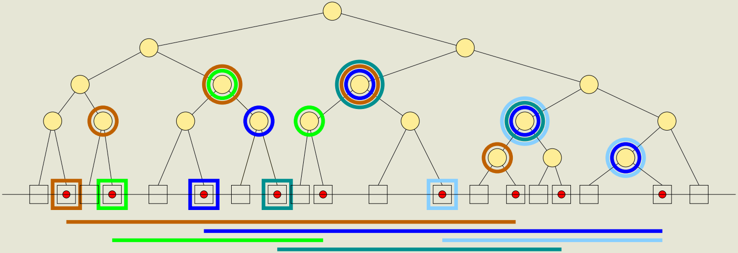
$$int(v) \subseteq [x, x'] \quad \text{and} \quad int(parent(v)) \not\subseteq [x, x']$$



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.



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

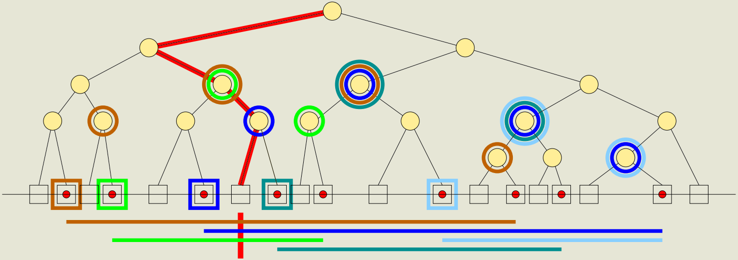
- build leaf-oriented balanced binary search tree on atomic intervals
- insert intervals one by one using InsertSegmentTree:

InsertSegmentTree$(v, [x, x'])$

if $\mathit{int}(v) \subseteq [x, x']$
    then add $[x, x']$ to $I(v)$
    else **if** $\mathit{int}(lc(v)) \cap [x, x'] \neq \emptyset$
            then InsertSegmentTree$(lc(v), [x, x'])$
            if $\mathit{int}(rc(v)) \cap [x, x'] \neq \emptyset$
            then InsertSegmentTree$(rc(v), [x, x'])$

QUERY

QUERY(q_x) reports all segments containing query point q_x.



QuerySegmentTree(v, q_x)

Report all the intervals in $I(v)$
if v is not a leaf
  then if $q_x \in int(lc(v))$
    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 $O(k + \log n)$.

Semi-Static Segment Tree

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.

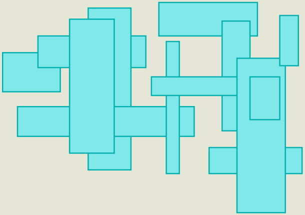
Insertion takes time $O(\log N)$.

Deletion of an interval s takes time $O(\log N)$ as well, if we know the positions of s in all sets $I(v)$ for all nodes v where s is stored.

Klee’s Measure Problem

What is the area of the union of a set of axis-aligned rectangles?

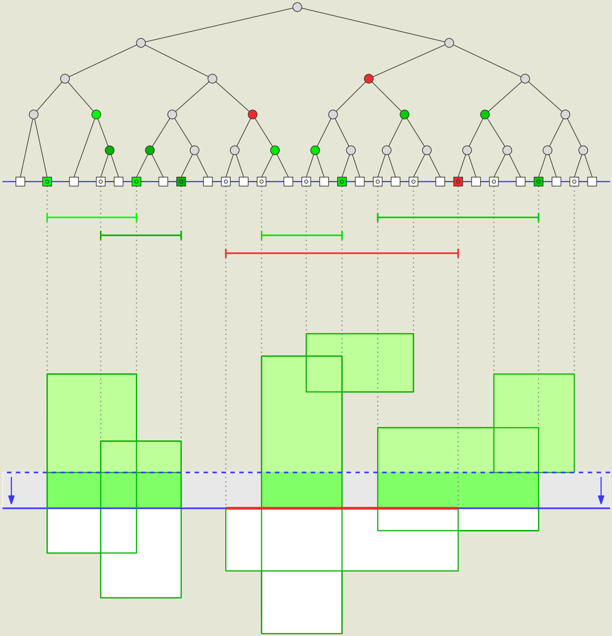
http://granmapa.cs.uiuc.edu/ jeffe/open/klee.html



Bentley's Algorithm

[Jon L. Bentley; *Algorithms for Klee's rectangle problems*, Unpublished notes, Computer Science Department, Carnegie Mellon University, 1977]

- use a plane-sweep algorithm
- maintain the intersection intervals of rectangles and sweep line
- maintain the length of the union of the intersection intervals
- use an augmented semi-static segment tree for that purpose



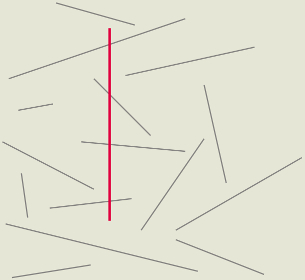
- all $2n$ interval endpoints are known in advance $\rightarrow$ semi-static
- for each node v in the semi-static segment tree we maintain the length of the union of the intersection intervals restricted to $int(v)$
- insert and delete operations in the augmented semi-static segment tree take time $O(\log n)$ each

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

Vertical Stabbing Queries for 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

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

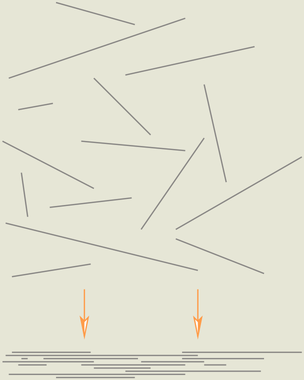


Augmented Segment Tree

for Vertical Stabbing Queries for Disjoint Line Segments

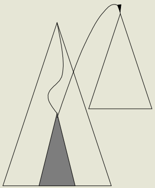
Underlying data structure:

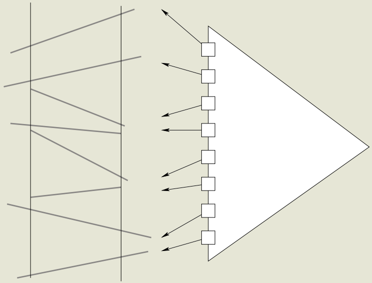
Basically, the underlying data structure is a segment tree for the projection intervals of the segments in S onto the x-axis. However, we don't store the intervals in $I(v)$ at a node v explicitly.



“Additional” information stored at each node v:

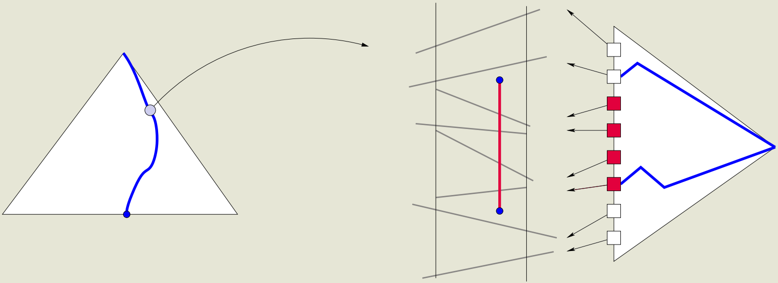
For a node v, let $S(v)$ be the set of segments corresponding to the intervals in $I(v)$. Store the segments in $S(v)$ in a leaf-oriented balanced binary search tree based on the order of the elements in the slab $int(v) \times (-\infty, \infty)$.





QUERY

$$\text{QUERY}(q_x \times [q_y, q_y'])$$



Search for q_x in the underlying segment tree. For each visited node v, search for q_y and q_y' in the balanced binary search tree for $S(v)$. Report segments which are between q_y and q_x' at q_x.

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 $O(n \log n)$.