Range Tree

Range Search 1-D

Given \( SC \subset \mathbb{R} \) with \( |S| = n \)
Query: interval \( I = [l, h] = \{ x \in S \mid l \leq x \leq h \} \)
Report: \( SN_I \)

Solution: Balanced binary search tree with all elements stored in leaves.
Link leaves from left to right. Search for \( I \). Report all leaves in the right \( \leq h \)
Called a 1-D range tree.

\( P(n) = O(\log n) \)
\( Q(n) = O(\log n + A) \)
\( S(n) = O(n) \)
\( A = \text{size of answer} \)

Range Search 2-D

Given \( SC \subset \mathbb{R}^2 \) with \( |S| = n \)
Query: rectangle \( R = [l_1, h_1] \times [l_2, h_2] \)
Report: \( SN_R \)

Solution: Naive

2 1-D range trees, one on x coord, one on y coord.
Find all pts in vertical range? take intersection
Find all pts in horizontal range?

\( P(n) = O(\log n) \)
\( Q(n) = O(\log n + |S_1| + |S_2|) \)
\( S(n) = O(n) \)

Wasteful

For all \( \binom{n+1}{2} \) possible vertical strips, with endpoints in \( S \),
create a 1-D range tree for the points on that strip.
For a query rectangle \( R \), do a 1-D query on widest vertical strip contained by extending \( R \) vertically.

\( P(n) = O(n^3) \)
\( Q(n) = O(\log n + A) \)
\( S(n) = O(n^3) \)
Decompose the plane into a small number of disjoint standard answer strips which correspond to subtrees of a balanced tree for S.

T balanced binary search tree for S
∀ ret, S_r = \{ p \mid p \in \text{subtree rooted at } r \}
strip_r = [T^r, h_r] \times \mathbb{R} = \text{vertical strip containing exactly } S_r.
Each possible answer strip \(\leq 2 \log n\) standard answer strips.
Each v has a 1-D range tree for \(S_v \cap \{x \leq x_v\}\).

if \(I_{1b} \cap h_r \cap I_{1d} \cap h_d \neq \emptyset\), then 1-D query \((C_l, b, I, I)\)
else
if \(C_{1b} \cap h_r \cap C_{1d} \cap h_d = \emptyset\), then return
2-D query \((C_{l1}, h_1) \times (C_{l2}, h_2)\) \(\leftarrow \text{rpt. sm} (v)\)

\(P_{1n} = \Theta (\log n)\)
\(O_{1n} = O (\log n + A)\) \(\downarrow\) later we will see a
\(S_{1n} = O (\log^d n)\) \(\downarrow\) way to reduce

Problem d-Dimensional Range Searching
Given \(S \subseteq \mathbb{R}^d, |S| = n\)
Query: rectangle \(R = [l_1, b_1, x_2, \ldots, x_d] \)
Report: \(R \cap S\)

Generalize the solution to 2-D range searching.
Balanced binary tree \(T\) on the \(x\) coordinate.
\(\forall r \in T\), create a secondary \((d-1)\text{-dim range tree}\)
on \([x_2, \ldots, x_d]\) cords.

\(P_{1n} = \Theta (n \log^{d-1} n)\) \(\downarrow\) can be improved
\(O_{1n} = O (\log^{d-1} n + A)\) to \(O (d-1 \log n + A)\)
Inverse Range Queries
Segment Trees
Interval Trees
Unbounded Rectangular Ranges and Trees

Last time, given a set of pts in \( \mathbb{R}^m \), constructed a data structure to report which points lay within a query range.

This time, given a set of ranges, construct a data structure to help us report in which of them a query point lies.

1-D Problem

Given a finite set \( S = \{ c_i, h_j \} \) \( i = 1 \ldots n \) \# line segments on the real line.

Query: \( x \in \mathbb{R} \)

Report: set of segments containing \( x \)

\( \forall c_i, h_j \in S \)

Segment Tree on \( S \) - intervals defined on endpoints

atomic intervals

first level

\( \left[ \left. c_i \right| \left. h_j \right] \)

2rt level

store all segments

Each node contains list of all segments covering it, but not its parent.

query \( (x, y) \)

if \( x \notin \mathbb{R} \) then return

report \( S_x \)

query \( (x, \text{child}(v)) \)

\( P(n) = C(n \log n) \)

\( S(n) = C(n \log n) \)

\( O(n) = O(\log n + A) \) when \( A \) is the size of the answer.
2-D Case

Given set of rectangles, each defined as the cross product of an $x$ and $y$ interval

$$R_i = [l_{x_i}, h_{x_i}] 	imes [l_{y_i}, h_{y_i}]$$

1-D segment tree on $x$-intervals

- Each node has a set $S_i$ containing rectangles
- Organize each $S_i$ into a segment tree on the $y$ intervals

Locate $x$-position of the query point in the primary segment tree
Locate $y$-position in the secondary segment tree at each vertex in which we pass.

Report up the rectangles found on the vertices of the secondary segment trees.

Query Time: $O((\log n + A))$


Only sort rectangles once along each coordinate

Space: $O(A \log^2 n)$

Each segment has $O(\log n)$ appearances in primary tree
Each causes $O(\log n)$ appearances up a secondary segment tree

Each $n \log^2 n$
Interval Trees

1-D unweighted range searching problem $\rightarrow$ more space efficient data structure

Given set of segments $S$

Given $pt \in \mathbb{R}$

Report segments containing $x$

Take endpoints of $S$ to multisets

Take median $m$ (may have multiple copies of endpoint)

Partition $S = S_{\leq m}, S_{> m}$

Report segments containing $m$

At node $v$ containing $m$

Store two lists: one of left endpoints of $S_v$, one of right endpoints of $S_v$

Answer query:

Use median values stored at each vertex to locate the query point as with binary search tree

At each node, search list from outside

Report segments containing it

$e(n) = \Theta(ln(n))$

$\ell(n) = \Theta(\log m + A)$

$g(n) = \Theta(mg) \rightarrow$ improvement in space.
2-D range query problem: query ranges are unbounded rectangles open on the top with sides extending to infinity. Found prune range trees to answer such queries but "treaps" hybrid pq trees and heaps use linear space.

McGeigh - Priority search trees.

Build a treap for a set \( S \subseteq \mathbb{R}^2 \) as follows:

0) If \( S = \emptyset \), do nothing.
1) Find \( p \in S \) with max \( y \) coord.
2) Remove \( p \) from \( S \).
   - Compute median \( y \) coord of remaining points.
   - Divide set into \( S_L \), \( S_R \) (left, right of median).
3) Make a vertex \( v \) with fields \((x, p)\) and \( m \),
   - left, right child pointers to treaps for \( S_L \), \( S_R \).

A treap has height \( O(\log n) \).
Every point is stored exactly once as the \( p \) field.
A vertex.
A treap is a heap as far as the \( y \)-coordinate
is concerned. With respect to the \( x \)-coordinate, it is "nearly" sorted in the sense that for a given path from root to leaf, the contents of all subtrees to the left of the path are less than the leaf node and the contents of those to the right are greater.

A query of the form \([lx, hx] \times [ly, \infty] \) is processed:

1) Locate the two leaves of the tree that would follow \( lx \)
an preceed \( hx \), respectively, branching on the median.
   - Values stored at the tree nodes.
   - For each node along the two traced root-leaf paths, report the point stored in the \( p \) field if it is contained
   - in the query range.
2) Search all subtrees between the two paths, reporting all points stored in the paths with y-coordinate not less than \( y_i \).

By the heap property, we can stop searching a path when we encounter a point with y-value below \( y_i \). So the vertices actually examined will form a forest of binary trees with \( A \) internal nodes, where \( A = \text{size of answer} \), hence containing \( O(A) \) total nodes. The entire query process takes time

\[
Q(n) = O(n \log n + k) \\
P(n) = O(n \log n) \\
S(n) = O(n)
\]
Problem
\( S = \text{set of intervals on } \mathbb{R} \)
Need data structure to answer:

\( a) \) given query interval \( I_0, \) return
\[ \{ I : I \cap I_0 \neq \emptyset \} \]
\[ \{ I : I \subset I_0 \} \]
\[ \{ I : I \subset I_0 \} \]

\( b) \)
\( c) \)

Solution
\( I \in S, I = [a, b] \) define
\[ S = \{ (b, a) \mid [a, b] \in S \} \] pts in \( \mathbb{R}^2 \)
\( I_0 = (x_0, y_0) \)

\( a) \) query \( S \) with open rectangle \((x_0, y_0, y_0)\)
- \((x_0, y_0, y_0)\) does it end after candidate?
- did it start before candidate ends?
- \( \text{i.e. all points } x_0 \leq b \leq \infty \text{ and } a \leq y_0 \)

\( b) \) open rectangle \((y_0, x_0, x_0)\)
- \((y_0, x_0, x_0)\) does it end before candidate ends
- \((y_0, x_0, x_0)\) does it start
- \( y_0 \leq b \leq \infty \text{ and } a \leq x_0 \)

\( c) \) \( S^* = \{ (a, b) \mid [a, b] \in S \} \)
query with open rectangle \((x_0, y_0, y_0)\)
- \( \text{i.e. all points } x_0 \leq a \leq y_0 \text{ and } b \leq y_0 \).
2-D Range Search

S = set of points in $\mathbb{R}^2$
Build a balanced binary search tree $T$ for $S$ wrt x-coord.
$T$ is primary data structure.

Let $T \subseteq T$. Define

$$\text{key}(v)$$
$$S_e = \{ p \in S | p \text{ stored in left subtree of } v \}$$
$$S_r = \{ p \in S | p \text{ stored in right subtree of } v \}$$

Build & condory data structure.

$$T_e = \text{priority search tree for } S_e \text{ to answer}$$

Each point occurs once at each level of the tree $T$

$$S(N) = O(N \log N)$$

Case 1)

do query in secondary structure

$$Q(n) = O(\log n + k) \text{ where } k \text{ is answer size}$$