Can we do better?

Range Trees:
- Space is $O(n \log n)$
- Query time: $O(\log n)$
  - Counting: $O(\log n)$
  - Reporting: $O(k \log n)$
- In $R^2$: $\log^2 n$ much better than $\log n$ for large $n$
- Range trees are more limited

Recap:
- kd-Tree: General-purpose data structure for pts in $\mathbb{R}^d$
- Orthogonal range query:
  - Count/report pts in axis-aligned rect. $O(n \log n)$
  - kd-Tree: Counting: $O(n)$ time
  - Reporting: $O(k \log n)$ time

Layering: Combining search structures
- Suppose you want to answer a composite query w. multiple criteria:
  - Medical data: Count subjects
    - Age range: $a_o \leq \text{age} \leq a_i$
    - Weight range: $w_o \leq \text{weight} \leq w_i$
- Design a data structure for each criterion individually
- Layer these structures together to answer full query
- Multi-Layer Data Structures

Claim: A 1-D range tree with $n$ pts has space $O(n)$ and answers 1-D range count/report queries in time $O(\log n)$ (or $O(k + \log n)$)

Call this a 1-Dim Range Tree:

Approach:
- Balanced BST (e.g. AVL, RB, …)
- Assume extended tree
- Each node $p$ stores no. of entries in subtree: $p.size$

Canonical Subsets:
- Goal: Express answer as disjoint union of subsets
- Method: Search for $Q_{i_0} + Q_{i_1}$

Range Trees I:
- Canonical subsets:
  - Goal: Express answer as disjoint union of subsets
  - Method: Search for $Q_{i_0} + Q_{i_1}$

Layering:
- Combining search structures
- Suppose you want to answer a composite query w. multiple criteria:
  - Medical data: Count subjects
    - Age range: $a_o \leq \text{age} \leq a_i$
    - Weight range: $w_o \leq \text{weight} \leq w_i$
- Design a data structure for each criterion individually
- Layer these structures together to answer full query
- Multi-Layer Data Structures
Recursive helper:
\[
\int \text{range}ID_x (\text{Node } p,
\begin{align*}
\text{Intv } Q &= [Q_h, Q_{hi}], \\
\text{Intv } C &= [x_0, x_1]
\end{align*}
\]
initial call: \text{range}ID_x(\text{root}, Q, C)

More details:
Given a 1-D range tree \(T\):
- Let \(Q = [Q_h, Q_{hi}]\) be query interval
- For each node \(p\), define interval cell \(C = [x_0, x_1]\)
  s.t. all pts of \(p\)'s subtree lie in \(C\)
- Root cell: \(C_0 = [-\infty, +\infty]\)

Cases:
- \(p\) is external:
  - if \(p.pt.x \in Q\) \(\rightarrow 1\) else \(\rightarrow 0\)
- \(p\) is internal:
  - \(C \subseteq Q\) \(\Rightarrow\) all of \(p\)'s pts lie within query
    \(\rightarrow\) return \(p.size\)
  - \(C\) is disjoint from \(Q\) \(\Rightarrow\) none of \(p\)'s pts lie in \(Q\)
    \(\rightarrow\) return \(0\)
  - Else partial overlap
    \(\rightarrow\) Recurse on \(p\)'s children + trim the cell

Range Trees II

2-D Range Searching:
- Layer a range tree for \(x\) with range tree for \(y\)
- For each node \(p \in 1D-x\) tree, let \(S(p) = \text{set of pts in } p\)'s subtree\)
- Def: \(p.txt\): A 1-D-y tree for \(S(p)\)

Analysis:

Lemma: Given a 1-D range tree with \(n\) pts, given any interval \(Q\), can compute \(O(\log n)\) subtrees of pts whose union is answer to query.

Thm: Given 1-D range tree...
can answer range queries in time \(O(\log n)\) \(\rightarrow\) \(k\) to report
Answering Queries?

Given query range

\[ Q = [Q_{lo,x}, Q_{hi,x}] \times [Q_{lo,y}, Q_{hi,y}] \]

- Run range 1Dx to find all subtrees that contribute
- For each such node p,
  - run range 1Dy on p.aux
- Return sum of all result

2D Range Tree:

- Construct 1D range tree based on x coord for all pts
- For each node p:
  - Let \( S(p) \) be pts of pi tree
  - Build 1D range tree for \( S(p) \) based on y \( \to \) p.aux
- Final structure is union of x-tree + (n-1) y-trees

Higher Dimensions?

- In d-dim space, we create d-layers
- Each recurses one dim lower until we reach 1-d search
- Time is the product:
  \[ \log n \cdot \log n \cdot ... \cdot \log n = O(\log^d n) \]

Analysis: The 1D x search takes \( O(\log n) \) time & generates \( O(\log^d n) \) calls to 1Dy search
\[ \Rightarrow \text{Total: } O(\log n \cdot \log n) = O(\log^2 n) \]

int range2D(Node p, Rect Q, Intv C=[x0,x1])

if (p is external) return p.pt \in Q ? 1
else if (Q.x contains C) \{ // C \subseteq Q\text{'s} x-projection
  [y0,y1] = [-\infty,\infty] // init y-cell
  return range1Dy(p.aux, Q, [y0,y1])
} else if (Q.x is disjoint of C) return 0
else
  \{ // partial x-overlap
    return range2D(p.left, Q, [x0, p.x])
    + range2D(p.right, Q, [p.x, x1])
  \}

Analysis:

Invoked \( O(\log n) \) times - once per maximal subtree
Invoked \( O(\log n) \) times - once for each ancestor of max subtree

Intuition: The x-layer finds subtrees p contained in x-range + each aux tree filters based on y.
Announcements: 11/08

- Program assignment 2 due Thu, 11:59 pm
  - Autograder is up

- HW 3 Preliminary is posted
  - Due in class next Tue
  - No late submissions

- Midterm 2 - Thu of next week
  - Closed book/notes
  - Allowed 2 cheat sheets/front and back

Total space: $\sum \text{space}_p = ?$

$n - 1$ int nodes each give rise to aux tree

Space:

$x$-tree

$\text{aux}$ tree

Total space:

$\sum 1.5(n) = ?$

Size of $\text{aux}$

$\frac{n}{m} \leq n \leq \frac{n}{m} \leq n$

$\Omega(n \log n)$