Can we do better?

Range Trees:
- Space is $O(n \log^d n)$
- Query time:
  - Counting: $O(\log^d n)$
  - Reporting: $O(k + \log^d n)$
- In $\mathbb{R}^2$: 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.
  - $kd$-Tree: Counting: $O(\log n)$ time
  - Reporting: $O(k + \log n)$ time
- Call this a 1-Dim Range Tree:

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 $O(k + \log n)$)

Layering: Combing search structures
- Suppose you want to answer a composite query w. multiple criteria:
- Medical data: Count subjects
  - Age range: $a_{i_0} \leq age \leq a_{i_d}$
  - Weight range: $w_{i_0} \leq weight \leq w_{i_d}$
- Design a data structure for each criterion individually
- Layer these structures together to answer full query

1-Dim Range Tree:
- $Q_{i_0}$ to $Q_{i_d}$
- 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 $Q_{i_0} + Q_{i_d}$ take maximal subtrees
Recursive helper:
\[ \text{int range1Dx(Node p, Intv Q=[Q_L, Q_R], Intv C=[x_L, x_R])} \]

Initial call: \( \text{range1Dx(root, Q, C_0)} \)

More details:
Given a 1-D range tree \( T \):
- Let \( Q=[Q_L, Q_R] \) be query interval
- For each node \( p \), define interval cell \( C=[x_L, x_R] \) s.t. all pts of \( p \)'s subtree lie in \( C \)

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

\[ \text{int range1Dx(Node p, Intv Q, Intv C=[x_L, x_R])} \] \( \{
\)
  if (\( p \) is external) \( \rightarrow 1 \)
  else if (\( C \subseteq Q \)) return \( p \) pivot \( x \in Q \) \( \rightarrow 0 \)
  else if (\( Q \cap C \) disjoint) return 0
  else return
    range1Dx\( p \).left, Q, [x_L, p pivot x]
    + range1Dx\( p \).right, Q, [p pivot x, x_R]
\[ \}

2-D Range Searching:
- Layer a range tree for \( x \) with range tree for \( y \)
- For each node \( p \) \( \in \) 1-D \( x \) tree, let \( S(p) \) = set of pts in \( p \) 's subtree
- Def: \( p \_aux \): 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 \( O(\log n) \) subtree whose union is answer to query.

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

- **Given query range** $Q = [Q_{lo,x}, Q_{hi,x}] \times [Q_{lo,y}, Q_{hi,y}]$
- **Run range1Dx** to find all subtrees that contribute
  - For each such node $p$:
    - Run range1Dy 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 $p$.tree
  - Build 1D range tree for $S(p)$ based on $y \rightarrow p$.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:
  $\log n \cdot \log n \cdots \log n = \mathcal{O}(\log^d n)$

**Analysis:** The 1D $x$ search takes
- of $\mathcal{O}(\log n)$ time & generates
- $\mathcal{O}(\log n)$ calls to 1D $y$ search
- $\Rightarrow$ Total: $\mathcal{O}(\log n \cdot \log n) = \mathcal{O}(\log^2 n)$

**Analysis:**

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

Intuition: The $x$-layer finds sub-trees $p$ contained in $x$-range & each aux tree filters based on $y$.