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
  $\mathbb{R}^d$ 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.
  - $\text{Ans} = 4$
- kd-Tree: Counting: $O(n)$ time
  - Reporting: $O(k + \log n)$ time

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: Combining search structures
- Suppose you want to answer a composite query w. multiple criteria:
  - Medical data: Count subjects w.:
    - Age range: $a_{i0} \leq \text{age} \leq a_{ih}$
    - Weight range: $w_{i0} \leq \text{weight} \leq w_{ih}$
  - Design a data structure for each criterion individually
  - Layer these structures together to answer full query
  → Multi-Layer Data Structures

1-Dim Range Tree:
- Canonical Subsets:
  - Goal: Express answer as disjoint union of subsets
  - Method: Search for $Q_{i0} + Q_{hi}$ + take maxima of subtrees
  - Approach:
    - Balanced BST (e.g. AVL, RB, ...)
    - Assume extended tree
    - Each node $p$ stores no. of entries in subtree: $p.size$

Call this a 1-Dim Range Tree:

1-Dim Range Tree:
- Canonical Subsets:
  - Goal: Express answer as disjoint union of subsets
  - Method: Search for $Q_{i0} + Q_{hi}$ + take maxima of subtrees
  - Approach:
    - Balanced BST (e.g. AVL, RB, ...)
    - Assume extended tree
    - Each node $p$ stores no. of entries in subtree: $p.size$
Recursive helper:
\[
\text{int range1Dx(Node p,}
\text{ Intv Q=[Q_l,Q_r], Intv C=[x_o,x_i])}
\]

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

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

Given a 1D range tree \(T\):
- Let \(Q=[Q_l,Q_r]\) be query interval
- For each node \(p\), define interval cell \(C=[x_0,x_i]\)
  s.t. all pts of \(p\)'s subtree lie in \(C\)
- Root cell: \(C_0=[-\infty,\infty]\)

Range Trees II

\[
\text{int range1Dx(Node p,}
\text{ Intv Q, Intv C=[x_0,x_i])}
\]

if (\(p\) is external) \(\rightarrow 1\)
\[
\text{if } (C \subseteq Q) \text{ return } p\text{.size}
\]
else if (\(Q+C\) disjoint) \(\text{return} 0\)
else return:

\[
\text{range1Dx(p.left, Q, [x_0, p.x])} + \text{range1Dx(p.right, Q, [p.x, x_i])}
\]

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

Analysis:

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

Thm: Given 1D 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 1D on $p.aux$
  - Return sum of all result

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

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 $y \mapsto p.aux$
- 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:
    \[
    \log n \cdot \log n \cdots \log n = O(\log^d n)
    \]

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

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

int range2D(Node p, Rect Q, Invtr C = [x0, x1])
if (p is external) return p.pt ∈ Q?
else if (Q.x contains C) \( \{ \)
  return range1Dy(p.aux, Q, [y0, y1])
else if (Q.x is disjoint of C) return 0
else
  return range2D(p.left, Q, [x0, p.x])
  + range2D(p.right, Q, [p.x, x1])