**Orthogonal Range Searching**

**Problem:** Preprocess a set of $n$ d-dimensional points, to support (axis aligned) d-dimensional rectangle queries.

---

**Variations**
- Preprocessing Time
- Query Time
- Space
- Dimension
- Static vs Dynamic point set
- Comparison model vs Integer coordinates
- Other queries: Count #points in region / Return point with max associated value / Return sum of points' associated values

---

1D
- Store points in search tree (elements at the leaves)
  - Preprocessing $O(n \cdot \log n)$
  - Space $O(n)$
  - Query $O(\log n + k)$

**kd-tree**
- Space $O(n)$
- Preprocessing $O(n \cdot \log n)$
- Query $O(\sqrt{n} + k)$

---

Report all subtrees between paths to $x_1$ and $x_2$

---

**height** $\log_2 n$
Kd-tree - Query Analysis

Nodes visited:

1) Node's rectangle completely contains query rectangle => At most \log n nodes
2) Node's rectangle contained in query rectangle => Complete subtree reported, i.e. change to k since k leaves reported and k internal nodes
3) Node partially overlaps with query rectangle (shaded area)
   a) Node or child stores a segment that is intersected by one of the \( k \) (infinite) lines defining the query rectangle.

Fact: Any horizontal/vertical line can at most be intersected by \( O(\sqrt{m}) \) nodes.

\[
\begin{align*}
\text{I}(n) & \leq 1 + 2I(\frac{n}{4}) \\
\text{I}(n) & \approx O(2^{\log_2 n}) = O(n^{\frac{1}{2}})
\end{align*}
\]

Total #nodes visited: \( O(n^{\frac{1}{2}} + k) \)

Note: Kd-trees can also support other (non-orthogonal) query-shapes, by recursive traversal of nodes intersecting query range/box at query range.

Range tree

- Nodes where subtree contains point within x-range
- \( \leq 2 \log n \) subtrees with points within range
- Store each set \( P(u) \) as a search tree w.r.t. y-coordinate

Queries: \( O(\log^2 n + k) \) - search in \( \leq 2 \log n \) \( T(u) \) trees
Space: \( O(n \cdot \log n) \) - each point stored in \( \log n \) \( T(u) \) trees at ancestors in \( T \)
Preprocessing: \( O(n \cdot \log n) \) - construct \( T(u) \) lists by merging siblings \( T^{(u)} \) lists
Higher dimensions

**kd-trees:** Space $O(n)$, Query $O(n^{1-\frac{1}{d}} + k)$, Preprocessing $O(n \cdot \log n)$
- Round-Robin split w.r.t. the d-1 levels
- Only every $d^{th}$ level is parallel w.r.t. to a side in query, i.e., only one child can contribute to the output

**Range trees:** Space $O(n \cdot \log n)$, Query $O(\log n + k)$, Preprocessing $O(n \cdot \log n)$
- Build $T$ on one dimension — each $T(v)$ structure is a range tree for d-1 dimensions
- Query: $T(d,n) = 2 \log n \cdot T(d-1,n)$, $T(1,n) = \log n$
  $\Rightarrow$ Space $S(d,n) \leq O(\log n)$, $S(1,n) = O(1)$

**Fractional cascading**

```
Goal: Search in $T(v_1)$ and $T(v_2)$ for $y_1$, $y_2$ but avoid using $O(\log n)$ time at each node

- Add links from each point in $T(v)$ to its immediate predecessor/successor w.r.t. y-value in both child lists
- Only need to search for $y_r$ of root in $O(\log n)$ time.
```

Space $O(n \cdot \log n)$, Query $O(\log n + k)$, Preprocessing $O(n \cdot \log n)$

**Summary of Results (2D)**

<table>
<thead>
<tr>
<th></th>
<th>Preprocessing</th>
<th>Space</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>kd-trees</strong></td>
<td>$O(n \cdot \log n)$</td>
<td>$O(n)$</td>
<td>$O(\sqrt{n} + k)$ *</td>
</tr>
<tr>
<td><strong>Range trees</strong></td>
<td>$O(n \cdot \log n)$</td>
<td>$O(n \cdot \log n)$</td>
<td>$O(\log n + k)$</td>
</tr>
<tr>
<td><strong>-t-Fractional cascading</strong></td>
<td>$O(n \cdot \log n)$</td>
<td>$O(n \cdot \log n)$</td>
<td>$O(\log n + k)$</td>
</tr>
<tr>
<td><strong>Chazelle</strong></td>
<td>$O(n \cdot \log n)$</td>
<td>$O\left(\frac{n \cdot \log n}{\log \log n}\right)$</td>
<td>$O(\log n + k)$ **</td>
</tr>
</tbody>
</table>

* Optimal query for $O(n)$ space
** Optimal space for fastest possible queries.