Orthogonal Range Searching

k-d tree

O(n) space
O(n logn) preproc time
query time O(√n) in 2D (+k report)
O(n^{1-2/d})

Better Method: Range Tree

divide by median x
store pts in sorted y-order←
recurse on left & right

Each node → vertical slab
Space: \( S(n) = 2S\left(\frac{n}{2}\right) + O(n) \)
\[ \Rightarrow \frac{O(n \log n)}{} \]

Preproc: \( P(n) = 2P\left(\frac{n}{2}\right) + O(n \log n) \)
\[ \Rightarrow \frac{O(n \log n)}{} \]

Query alg, given query vec \( \mathbf{q} \):

// counting

if (\( \mathbf{q} \) doesn't intersect node's slab)
    return 0

if (\( \mathbf{q} \) cuts completely across slab)
    do binary search in \( y \)-sorted lists
    recurse on both children
    return sum

visit \( \leq 2 \log n \) nodes
at each node, spend \( O(\log n) \) time for binary search

\[ \Rightarrow \text{query time } O(\log^2 n) \]

(dynamic: insert/delete \( O(\log^2 n) \))

Improving query time:

idea - add pointers from parent list to child list
query time $\mathcal{O}(\log n + \log n) = \mathcal{O}(\log n)$

Higher-D:

3D range tree

$S_d(n) = 2S_{d-1}(n) + S_{d-1}(n)$
$\Rightarrow S_d(n) = \mathcal{O}(S_{d-1}(n) \log n)$
$\Rightarrow \mathcal{O}(n \log^{d-1} n)$

$Q_d(n) = \mathcal{O}(Q_{d-1}(n) \log n)$
$\Rightarrow \mathcal{O}(\log^{d-1} n)$

Remark - trade-offs via degree-b range tree...

Improving space for 2D counting: (Chatelle '88)

$O(n)$ space, $O(\log n)$ query time

Idea - "bit packing"
at each node,
replace \( y \)-sorted list of points/pointers
by list of bits

\[ \Rightarrow \text{"compact range tree"} \]

\[
\begin{array}{c}
\phi, \\
\text{Pz} \\
\text{Pz} \\
\phi
\end{array}
\]

\[
\begin{array}{c}
\text{Pz} \\
\text{Pz} \\
\text{Pz}
\end{array}
\]

\[
\begin{array}{c}
\text{Pz} \\
\text{Pz} \\
\phi
\end{array}
\]

Left \( \phi \), right \( \text{Pz} \)

**Assumption**
- Standard word RAM model
  where each word is \( w \)-bit long
  with \( w \geq \log n \) (std. ptr/index fits in a word)

**Subproblem**
store string \( s \) of \( n \) bits
such we can answer rank queries:
given \( i \), compute \( \text{rank}_0(i) = \# 0's \) in \( s[1..i] \)

\[
\begin{array}{c}
101101110110110
\end{array}
\]

\[ \text{rank}_0(9) = 3 \]

**trivial:** \( \Theta(n) \) space, \( \Theta(1) \) time

**Q:** \( \Theta(n) \) bits of space? Yes... next time...