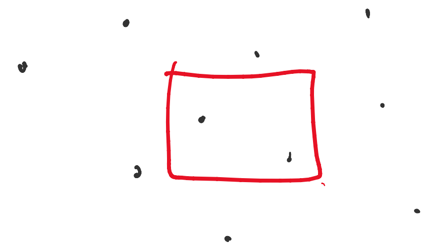
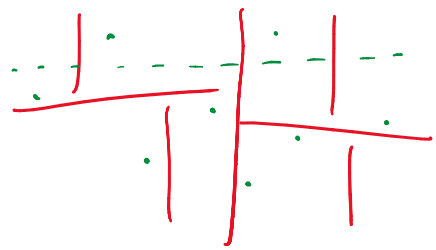
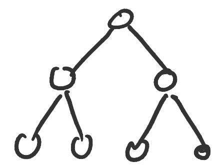
August 26, 2021 10:15 AM

Orthogonal Range Searching



k-d tree





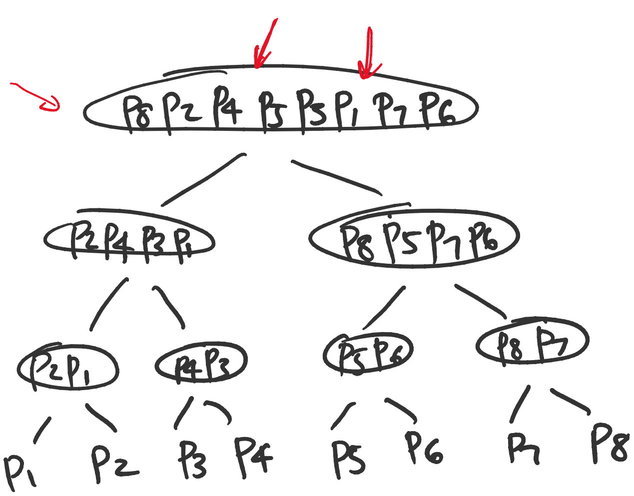
$f(n) = 2f(\frac{n}{4})$

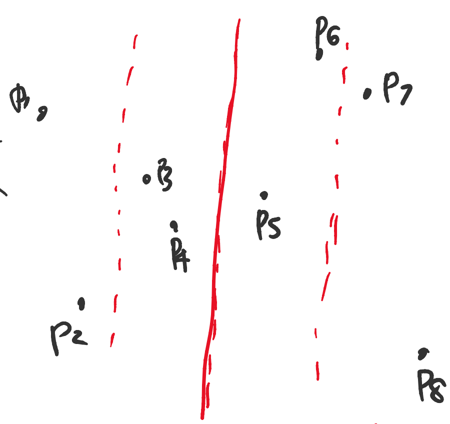
$O(n)$ space
$O(n \log n)$ preproc time
query time $O(\sqrt{n})$ in 2D
(+k report)

$O(n^{1-1/d})$

Better Method: Range Tree

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





each node → vertical slab

Space: $S(n) = 2S(\frac{n}{2}) + O(n)$

$\Rightarrow \boxed{O(n\log n)}$

Preproc: $P(n) = 2P(\frac{n}{2}) + O(n \cancel{\log n})$ by pre-sorting

$\Rightarrow \boxed{O(n\log^2 n)}$

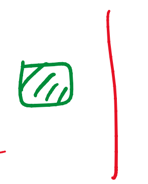
Query algn, given query rect q: //counting

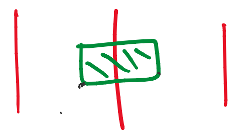
if (q doesn't intersect node's slab)
    return 0

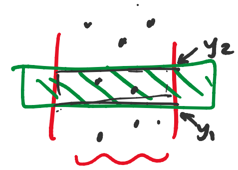
if (q cuts completely across slab)
    do binary search in y-sorted lists & return ans ...

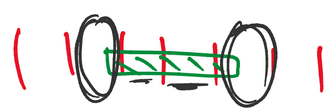
recurse on both children

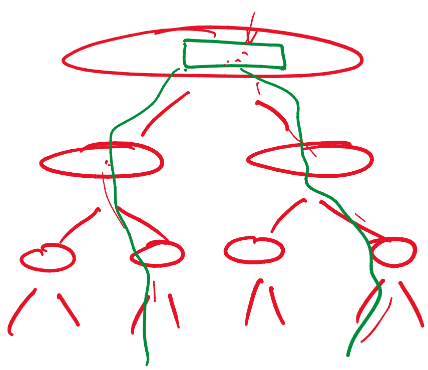
return sum











visit $\leq$ 2 log n nodes (≤ 2 nodes per level)

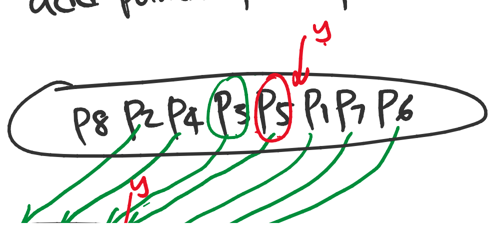
at each node, spend $O(\log n)$ time for binary search

$\Rightarrow$ query time $\boxed{O(\log^2 n)}$

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

Improving query time:

idea - add pointers from parent list to child list



only need 1 binary search in root list

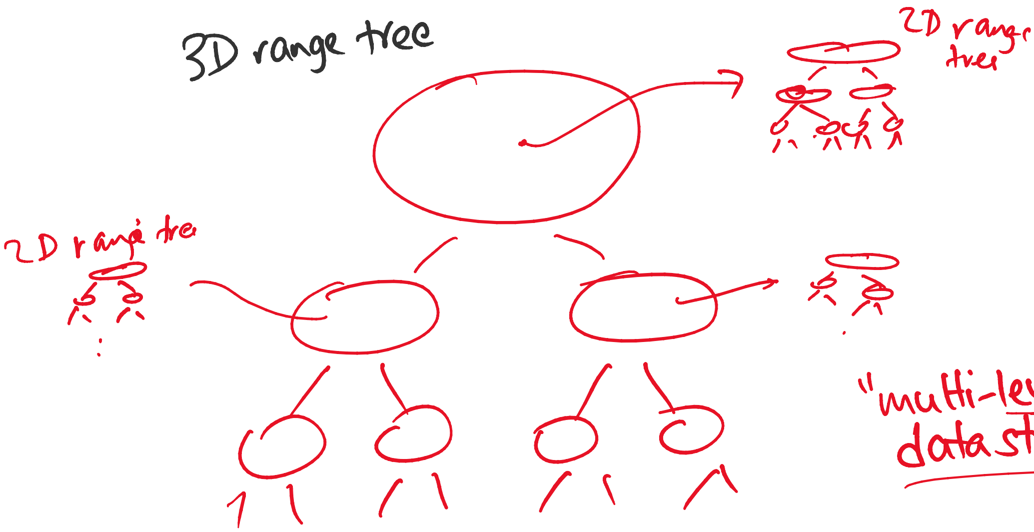
1 binary search at root list

$\Rightarrow$ query time $O(\log n + \log n)$

$= \boxed{O(\log n)}$ (+k for report)

Higher-D:

3D range tree



"multi-level data structure"

$$S_d(n) = 2\,S_d\left(\tfrac{n}{2}\right) + S_{d-1}(n)$$

$$\Rightarrow \quad S_d(n) = O(S_{d-1}(n)\log n)$$

$$\Rightarrow \quad \boxed{O(n \log^{d-1} n)}$$

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

$$\Rightarrow \quad \boxed{O(\log^{d-1} n)}$$

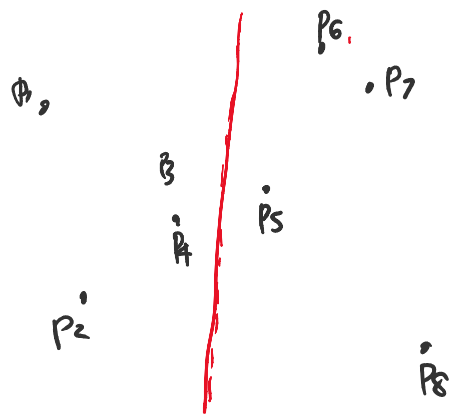
Rmk - trade-offs via degree-b range tree...

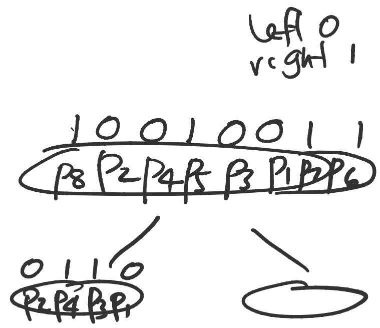
Improving space for 2D counting: (Chazelle '88)

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

idea - "bit packing"

at each node,
replace y-sorted list of points/pointers
by list of bits
$\Rightarrow$ "compact range tree"





assumption - standard word RAM model
where each word is w-bit long
with $w \geq \log n$ (s.t. ptr/index fits in a word)

Subproblem store string s of n bits
s.t. we can answer rank queries:
given i, compute $\text{rank}_0(i) = \#$ 0's in $s[1..i]$

1 0 1 1 0 1 1 1 0 1 1 0 1 1 0

$\text{rank}_0(9) = 3$

trivial: $O(n)$ space, $O(1)$ time

Q: $O(n)$ bits of space? Yes... next time...