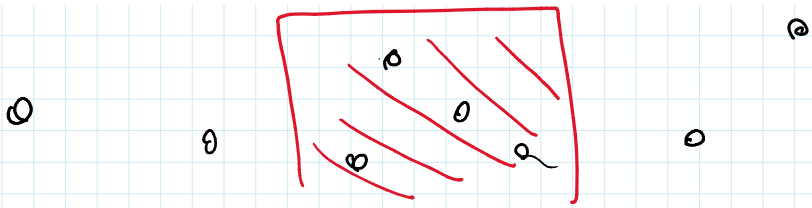


reduces to 1D range min query

$\Rightarrow$ $O(n)$ space

query time $O(\log n + 1) = \boxed{O(\log n)}$

for report-one

2 pred search in x

for report-all: $\boxed{O(\log n + k)}$

by repeatedly find min & recurse left & right

2D 4-sided?

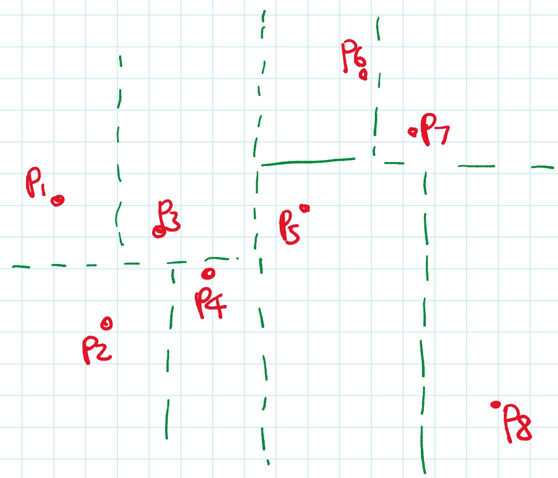
Method 1: k-d Tree

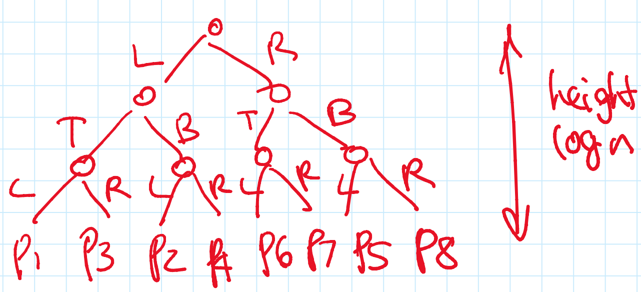
divide by median x

then median y

then median x

⋮





each node corresponds to a rectangular cell

Space $\boxed{O(n)}$

preproc time $P(n) = 2P(n/2) + O(n)$

preproc time $P(n) = 2P(n/2) + O(n)$

by median find

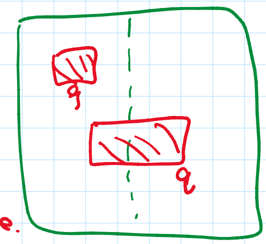
$\Rightarrow \boxed{O(n \log n)}$

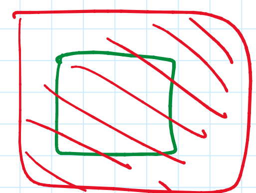
query alg/m, given rectangle q : // counting

→ if q does not intersect node's cell
  return 0

→ else if q completely contain cell
  return # pts in cell ← can precompute.

recurse in both children
return sum





Analysis :

query time = O(# cells visited)

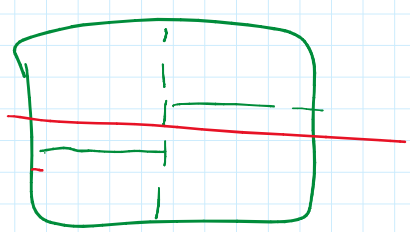
= O(# cells intersected by ∂q)

boundary of q

Consists of 4 line segments
vertical or horizontal

$\cancel{Q(n) = 2Q(\tfrac{n}{2}) + O(1)}$

Let $f(n)$ = max # cells intersected by
a vertical/horizontal line



$$f(n) \le 2f\left(\frac{n}{4}\right) + O(1)$$

$$\Rightarrow O(n^{\log_4 2}) = O(\sqrt{n})$$

$T(n) = aT(\frac{n}{b}) + \ldots$

$n^{\log_b a}$

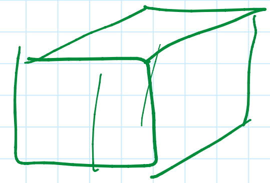
$n^{\log_b 4}$

$\Rightarrow$ query time $O(4 \cdot f(n)) = \boxed{O(\sqrt{n})}$ for counting

(or $O(\sqrt{n} + k)$ for reporting)

3D: $f(n) \le 4 f(\frac{n}{8}) + O(1)$

$\Rightarrow O(n^{2/3})$

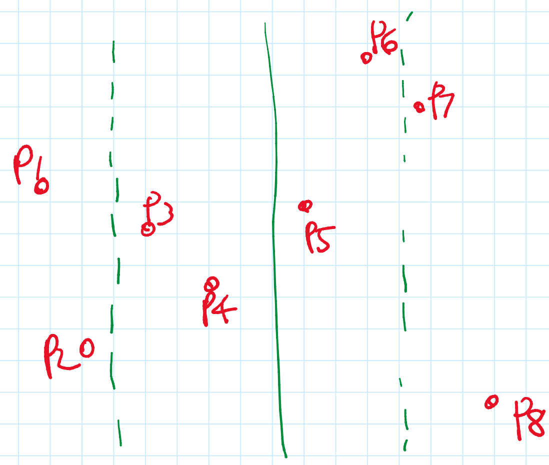


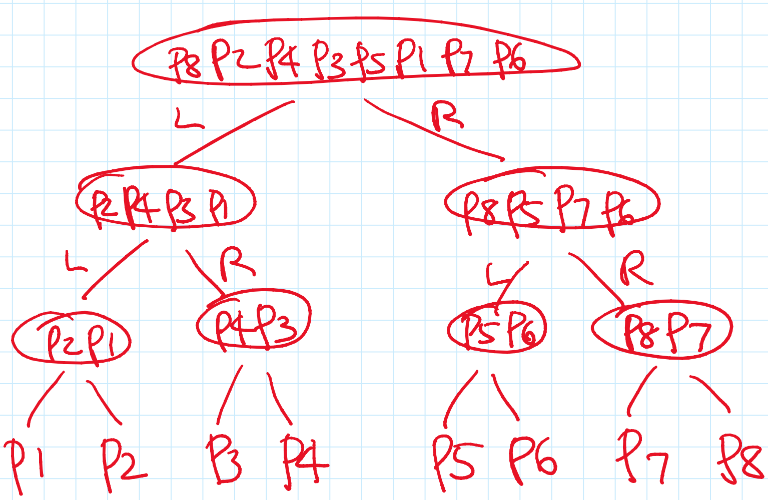
$\vdots$

Higher-D: $O(n^{\frac{d-1}{d}})$

Method 2: Range Tree

divide by median x
store pts in y-sorted list
recourse on left & right





each node corresponds to a vertical slab

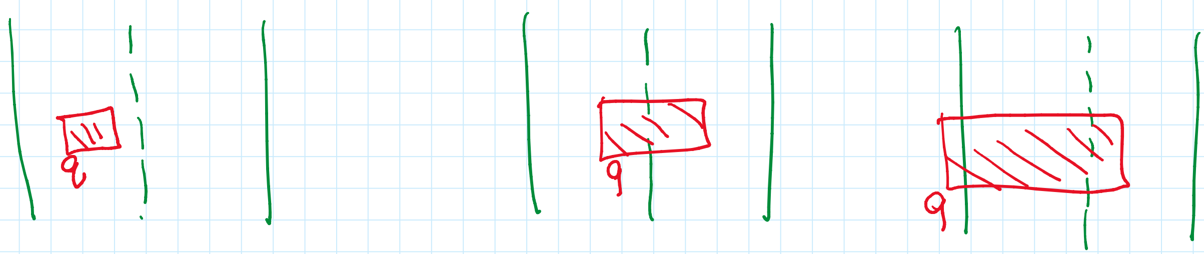
Space $\quad S(n) = 2S(\frac{n}{2}) + O(n) \Rightarrow \boxed{O(n\log n)}$

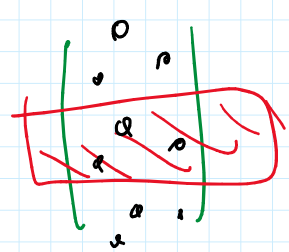
preproc time $\quad P(n) = 2P(\frac{n}{2}) + O(n\cancel{\log n}) \Rightarrow \boxed{O(n\log^{\cancel{2}} n)}$

by pre-sorting

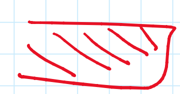
query algm, given rectangle q: // counting

if q does not intersect node's slab
    return 0
if q completely cuts across slab
    do binary search on y-sorted list
recurse in both children
return sum

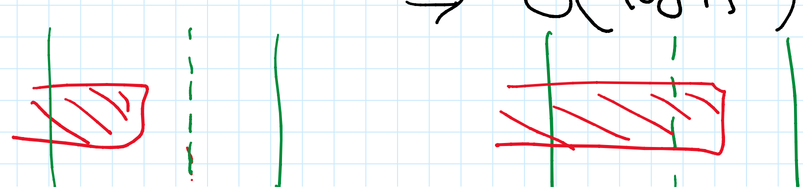


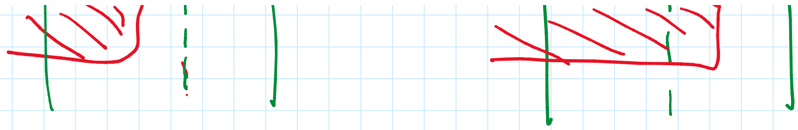


Analysis for 3-sided queries:



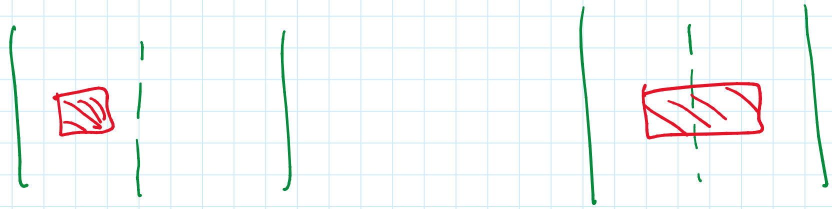
$$Q_3(n) \le 1\, Q_3\left(\frac{n}{2}\right) + O(\log n)$$
$$\Rightarrow O(\log^2 n)$$





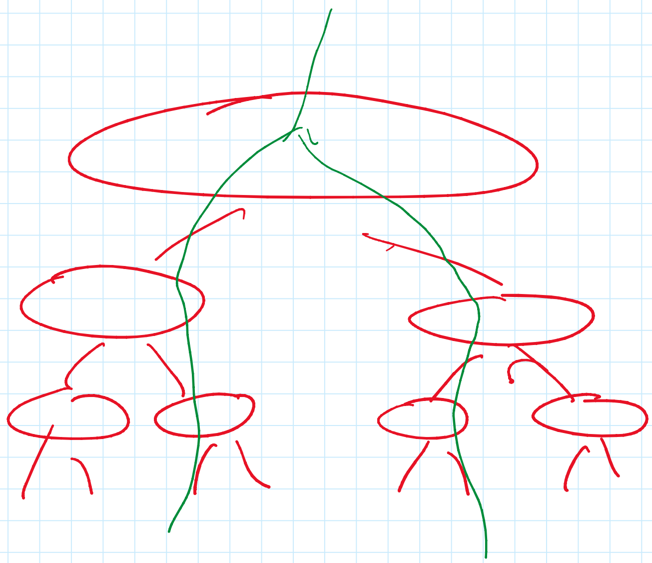
Analysis for general 4-sided:

$$Q_4(n) \le \max \begin{cases} 1\, Q_4(n/2) + O(1) \\ \cancel{2\, Q_3(n/2) + O(1)} \quad O(\log^2 n) \end{cases}$$



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

(+k for reporting)



$O(2 \cdot \log n)$

binary search at nodes along 2 paths

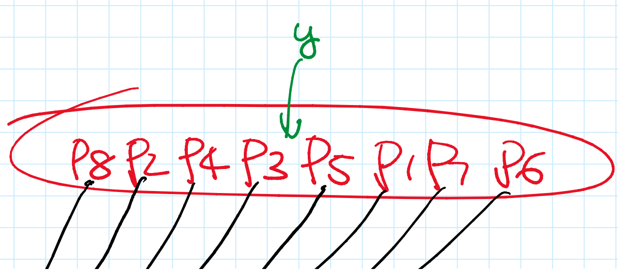
$\log n$ per binary search

$\Downarrow$

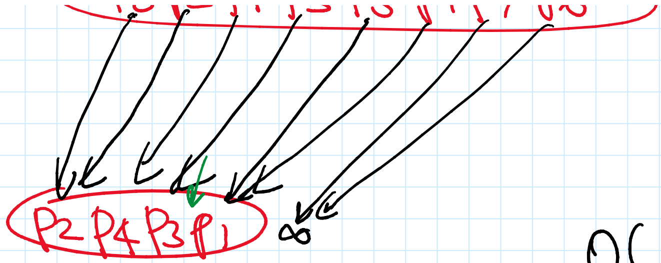
$O(\log^2 n)$

In static case:

can reduce query time to $\boxed{O(\log n)}$ (+k for reporting)



Store succ ptrs from parent list



store succ ptrs from parent list to child list

$$O(\log n + (\log n) \cdot 1)$$

init binary search

const time per node along path

$$= O(\log n)$$

dynamic?

replace y-sorted list with BST
use weight-balancing for overall tree

insert/delete $O(\log^2 n)$ amortized
query $O(\log^2 n)$