reduces to 1D range min query

\[ \Rightarrow O(n) \text{ space} \]

query time \( O(\log n + 1) \)

for report-one

\[ \begin{align*}
&\text{search in } x \\
&\text{Pred} \\
&\text{by repeatedly find min}
\end{align*} \]

for report-all: \( O(\log n + k) \)

2D 4 Sided?

Method 1: k-d Tree

divide by median \( x \)

then median \( y \)

then median \( x \)

Space \( O(n) \)

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

each node corresponds to a rectangular cell
preproc time \( P(n) = 2 P(n/2) + O(n) \)

by median-freq

\[ \Rightarrow O(n \log n) \]

query algm, given rectangle \( q \):
// counting

- if \( q \) does not intersect node's cell
  return 0
- else if \( q \) completely contains cell
  return #pts in cell
  recurse in both children
  return sum

Analysis:
query time = \( O(\# \text{cells visited}) \)
= \( O(\# \text{cells intersected by } q) \)

boundary of \( q \)
\( q \)
Consists of 4 line segments vertical or horizontal

Let \( f(n) = \max \# \text{cells intersected by} \)

a vertical/horizontal line

\( f(n) \leq 2 f(n/4) + O(1) \)

\[ \Rightarrow O(n \log_4^2) = O(\sqrt{n}) \]
\[ n^{\log_2 3} \]

\[ \Rightarrow \text{query time } O(4 \cdot f(n)) = \begin{cases} O(n) & \text{for counting} \\ (or } O(\sqrt{n} + k) \text{ for reporting) } \end{cases} \]

3D:
\[ f(n) \leq 4 \cdot f\left(\frac{n}{8}\right) + O(1) \]
\[ \Rightarrow O(n^{2/3}) \]

Higher-D:
\[ O\left(n^{\frac{d+1}{d}}\right) \]

**Method 2: Range Tree**

Divide by median x

Store pts in y-sorted list

Recurse on left & right

Each node corresponds to a vertical slab
Space \( S(n) = 2S\left(\frac{n}{2}\right) + O(n) \Rightarrow O(n \log n) \)
Preproc time \( P(n) = 2P\left(\frac{n}{2}\right) + O(n \log n) \Rightarrow O(n \log n) \)

by pre-sorting

query alg (y 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:

\[ Q_3(n) \leq 1 \cdot Q_3\left(\frac{n}{2}\right) + O\left(\log^2 n\right) \Rightarrow O\left(\log^2 n\right) \]
Analysis for general 4-sided:

\[ O_4(n) \leq \max \left\{ \frac{1}{2} O_4(\sqrt{2}) + O(1), \frac{3}{2} O_4(\sqrt{2}) + O(1) \right\} = O(\log^2 n) \]

\[ \Rightarrow O(\log n + \log^2 n) = O(\log^2 n) \quad (\text{for reporting}) \]

In static case:

can reduce query time to \( O(\log n) \) (\( \log n \) per binary search)

\( O(2\log n) \)

store succ pts

binary search at nodes along 2 paths

(\( \log^2 n \))

\( (\text{tk for reporting}) \)
Store next pointers from parent list to child list

\[ O(\log n + (\log n) \cdot 1) \]

init binary search

= \( O(\log n) \)

Dynamic?

replace ysorted list with BST
use weight-balancing for overall tree

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