kd-Trees

Build balanced in O(n log n) preprocessing sort by x to get list X
sort by y to get list Y
build (t, x, X, Y)

build (x, y)

Find the median in the cutting dimension
make the cut of the current subtree
split X into Xsubtree, Xleft
split Y into Ysubtree, Yleft
build (t → left, Xsubtree, Ysubtree)
build (t → right, Xleft, Yleft)

\[ X = \{ K, C, D, H, B, A, J, F, E, I \} \]
\[ Y = \{ J, B, C, E, A, D, H, F, G, I \} \]

nearest (n, p, nearest, d)
if n == NULL or p == n then return
if closest dist from p to origin bounding box subtree ≥ d return
if \[ |p - n → point| < d \] nearest = \[ n → point \] \[ d = |p - n → point| \]
if p on left nearest (n → left, p, nearest, d)
else nearest (n → right, p, nearest, d)
if p on right nearest (n → right, p, nearest, d)
else nearest (n → left, p, nearest, d)
\[ x \leq 10 \quad \text{and} \quad y \leq 10 \]

\[ n \leq c \cdot n \quad \text{and} \quad n \leq c \cdot n \]

\[ \frac{n}{2} \leq c \cdot n \quad \text{and} \quad \frac{n}{2} \leq c \cdot n \]

\[ \frac{n}{4} \leq c \cdot n \quad \text{and} \quad \frac{n}{4} \leq c \cdot n \]

\[ \frac{n}{8} \leq c \cdot n \quad \text{and} \quad \frac{n}{8} \leq c \cdot n \]

\[ \frac{n}{16} \leq c \cdot n \quad \text{and} \quad \frac{n}{16} \leq c \cdot n \]

\[ \frac{n}{32} \leq c \cdot n \quad \text{and} \quad \frac{n}{32} \leq c \cdot n \]

\[ \frac{n}{64} \leq c \cdot n \quad \text{and} \quad \frac{n}{64} \leq c \cdot n \]

\[ \frac{n}{128} \leq c \cdot n \quad \text{and} \quad \frac{n}{128} \leq c \cdot n \]

\[ \frac{n}{256} \leq c \cdot n \quad \text{and} \quad \frac{n}{256} \leq c \cdot n \]

\[ \frac{n}{512} \leq c \cdot n \quad \text{and} \quad \frac{n}{512} \leq c \cdot n \]

\[ \frac{n}{1024} \leq c \cdot n \quad \text{and} \quad \frac{n}{1024} \leq c \cdot n \]

\[ \frac{n}{2048} \leq c \cdot n \quad \text{and} \quad \frac{n}{2048} \leq c \cdot n \]

TOTAL \( \leq c \cdot n \cdot \log_2 n \)

\[ = c \cdot n \cdot \log_2 n \]

\[ \leq 0 \cdot (n \log n) \]
Red-Black Trees

Binary Search Trees such that

1) each node has a color: red or black

2) root is colored black

3) all leaves are colored black

4) children of red nodes are black

5) every path root → leaf has same # black nodes

---

[Diagram of Red-Black Trees]

Not a red-black tree

→ 0 or 2 children per node

# leaves in a non-empty full binary tree = # internal nodes + 1 (non-leaves)

if height is \( O(\log_2 \text{total nodes}) = O(\log_2 \text{data nodes} + 1) \)

\( = O(\log_2 \text{data nodes}) \)

(black height)

suppose # black nodes in path root → leaf is

shortest possible path has nodes

longest possible path has

Red-black tree with black height 5 has \( \geq 2^5 - 1 \) nodes

\( n \geq 2^5 - 1 \)
1) Do normal BST insert, color new node red (w/black laws)

2) If parent of new node exists and is black > DONE

Else

let z = new node

while z is not root and z.parent -> color = red

if parent is red and sibling exists and is red

let y = sibling

let z.parent -> color = black

let sibling -> color = black

let parent -> color = red

z.sibling -> left = z

z.sibling -> right = z.parent

z.parent -> left = z.sibling

z = parent

else

if z is left child

let c = z.sibling

let z.parent -> color = red

let c -> color = black

let z.sibling -> left = z

let z.sibling -> right = c

z = z.parent

else

let c = z.sibling

let z.parent -> color = red

let c -> color = black

let z.sibling -> left = c

let z.sibling -> right = z

z = z.parent

z = parent

[other cases symmetric]

DONE

e -> root -> color = black