Orthgonal Range Searching

11/20/08

BKOS6 Chap5

Problem report all people $20 \le age \le 30$

Query $100 \le weight \le 130$

$\$1K \le Salary \le \$3K$

Insert
Delete
Query

Chap 5 only handle case Insert ... Insert Query ... Query

ie Preprocess points then answer queries

$P = \{P_1 \cdots P_n\}$ $[x, x', y, y', z, z']$

1-D Range Search

A) Simplest solution

1) sort P and place into an array
2) Query $= [x, x']$
3) use binary search to find smallest P_i st $x \le P_i$
4) While $P_i \le x'$ report P_i; $i+1 = i$

Analysis

1) $O(n \log n)$ preprocessing time

2) $O(\log n + K)$ query processing time where $K = \#$ hits

B) Use any Balanced BST

Two natural trees

1) root is pivot p

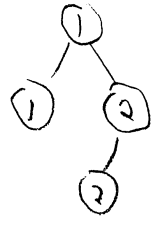
   1) $LST = \{q < p\}$ n node tree
      $RST = \{q > p\}$

---

   2) $LST = \{q \leq p\}$ $\approx 2n$ nodes
      $RST = \{q > p\}$

BKDS

every point in a leaf



Case 1

Report interval (T, I, x)

if x is a leaf & $x_v \in I$ report x_v

else if $x \in I$ then $RI(T, I, LC(x))$, $RI(T, I, RC(x))$

else if $I < x_v$ $RI(T, I, LC(x))$

else $I > x_v$ $RI(T, I, RC(x))$

Report Interval (T: Tree, I: $[a,b]$, x: node)

if x is empty return

if x is leaf & $x_v \in I$ report V_v

else $a \leq x_v$ then RI(T, I, LC(x))

$x_v < b$ then RI(T, I, RC(x))

Thm 1D range query

1) $O(n \log n)$ preprocessing
2) $O(n)$ storage
3) $O(\log n + k)$ reporting time

Pf Charging Rule

1) If a leaf & in range charge node
2) If both subtrees have nodes in range charge children
3) If at most one subtree has nodes in range charge node.

Kd-trees

For 2d Range Query

Input: Intervals I_x & I_y

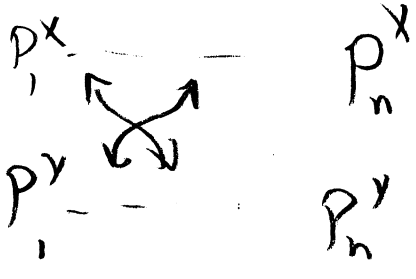
n points (no 2 have same x or y value)

Construction: Alternately split on median x-value or y-value.

Cost of medians

Construction 1) Using linear time median alg.

a) Sort P by x & y value & cross link

eg 

Split on x $p_1^x \ldots p_{n/2}^x \; p_{n/2+1}^x \ldots p_n^x$) $O(n)$ time

"Unshuffle"

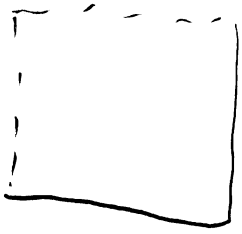
Def Region of a node X

Regin(Root) = Plane

$R = Regin(X)$

$R = R_1 \vee R_2 \cup Line$ $\quad R_1 = Regin(LC(X))$

$R_2 =$



Alg SearchKdTree(V, R)

1) $V_p \in R$ report V

~~1) if V is leaf then report V of $V_p \in R$~~

2) if $Regin(V) \subseteq R$ then ReportSubtree(V)

3) For each child C of V

if $Regin(C) \cap R \neq \emptyset$ then SearchKdtree(C, R)

Lemma Kd [illegible] Reporting time is

$O(\sqrt{n}+k)$

---

we need to count # of reached rectangles R'

$R' \subseteq R$ change to $\subset$

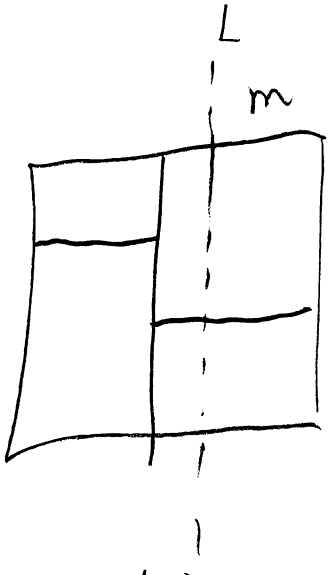
$R' \cap R \neq \emptyset$

1) we need to count # of rect intersects boundary

L = vertical line segment

$Q(n)$ = # rect of size n intersect L

$Q(n) = 1$



$Q(n) \leq 2Q(n/4)$

$Q(n) = O(\sqrt{n})$

Range Tree

Goal $O(\log^2 n + K)$ reporting time

$O(n \log n)$ storage

Data structure

1) Build 2d, say in x, Bal BST T_x

2) ~~$\forall$~~ for each $V \in T_x$

Build Bal BST in y for $P \in$ Subtree (V, T_x)

~~T_y^V~~ T_y^V

Range Search (T, R)

1) find $O(\log n)$ nodes N of T_x covering the Range$_x$ of R

2) $\forall V \in N$ search T_y^V for points in R_x

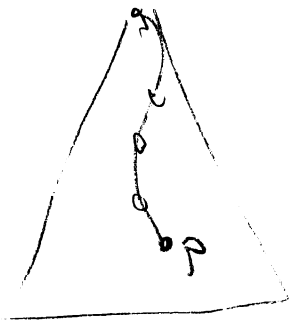
Query Search ~~time~~ Report Time

$$\sum_{V \in N} C(\log n + k_V) \qquad \sum k_V = K$$

$$\leq O(\log^2 n + K) \qquad |N| = O(\log n)$$

Claim Range Tree is of size $O(n \log n)$

$p \in P$



p is only in T_y^{δ} for δ an ancestor of p

$$\sum size(T_y^p) = O(n \log n)$$

the topmost array is associated with the root, the left array below it is associated with the left child of the root, and so on. Not all pointers are shown in the figure.)

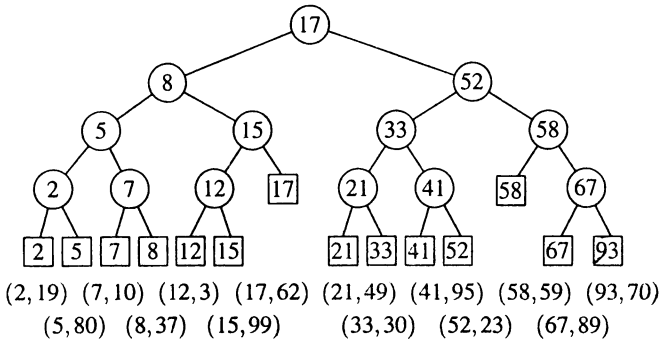


Figure 5.8
The main tree of a layered range tree: the leaves show only the x-coordinates; the points stored are given below

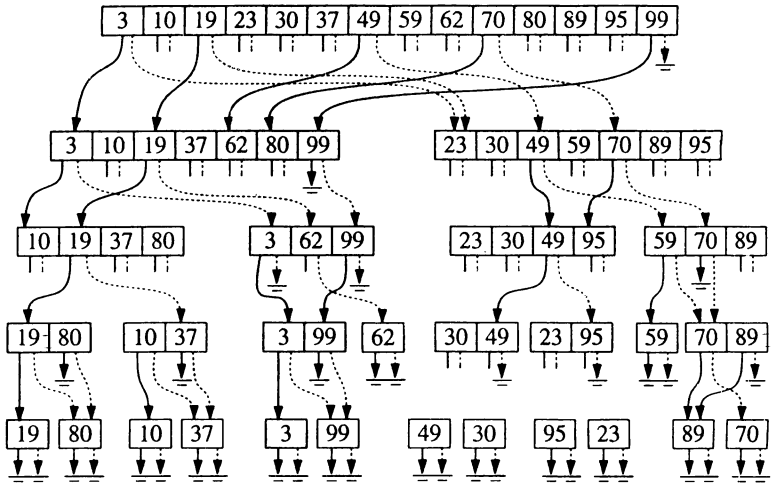
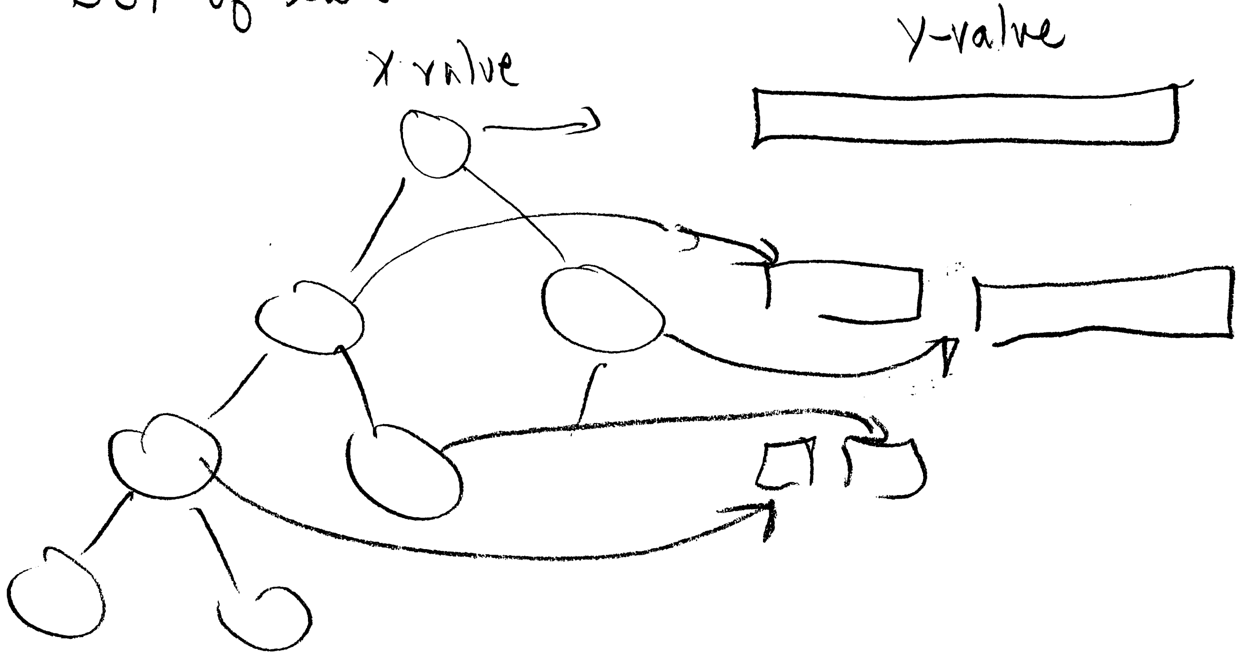


Figure 5.9
The arrays associated with the nodes in the main tree, with the y-coordinate of the points of the canonical subsets in sorted order (not all pointers are shown)

Let's see how to answer a query with a range $[x : x'] \times [y : y']$ in a layered range tree. As before we search with x and x' in the main tree $\mathcal{T}$ to determine $O(\log n)$ nodes whose canonical subsets together contain the points with x-coordinate in the range $[x : x']$. These nodes are found as follows. Let v_{split} be the node where the two search paths split. The nodes that we are looking for are the ones below v_{split} that are the right child of a node on the search path to x where the path goes left, or the left child of a node on the search path to x' where the path goes right. At v_{split} we find the entry in $A(v_{split})$ whose y-coordinate is the smallest one larger than or equal to y. This can be done in $O(\log n)$ time by binary search. While we search further with x and x' in the main tree, we keep track of the entry in the associated arrays whose y-coordinate is the smallest one larger than or equal to y. They can be maintained in constant time by following the pointers stored in the arrays. Now let v be one of the $O(\log n)$ nodes we selected. We must report the points stored in $A(v)$ whose y-coordinate is in the range $[y : y']$. For this it suffices to be able to find

Fractional Cascading

Idea BST of lists



$O(\lg n + k)$ per query

$O(n \lg n)$