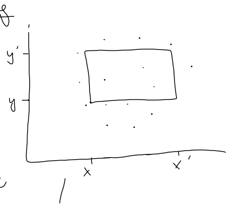
Orthogonal range searching

- Consider a set $P \subseteq \mathbb{R}^d$ & $[x_1,x_1'] \times \ldots [x_d,x_d'] \subseteq \mathbb{R}^d$ a range, Find points of P belonging to the range.
- Relevant to querying database



2-d case

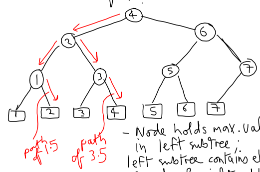
lis 8-15:50

1-d range searching

- $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}$ & real numbers $x \leq x'$.
- Points of P will be stored in balanced binary search tree, whose leaves are elements of P.
- Given $x \in \mathbb{R}$, it determines a path from root to a leaf: at a node v, go left if $x \leq x_v$ & right if $x_v < x$.

  $x = 1.5$, $x' = 3.5$.
- Given $x \leq x'$ the split node is the last node at which paths for x & x' coincide.

  Splitnode(1.5, 3.5) = 2.

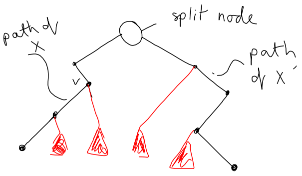


- Node holds max. value in left subtree; left subtree contains elts ≤ node & right subtree strictly larger elements.

lis 8-16:07

Algorithm: - given set of points P stored in bal. bin tree T & $x \leq x'$.
- Find points of P in range $x \leq x'$.

- Find split node v_{split}.
  - If split node is a leaf, check if leaf belongs to $[x, x']$
- Follow path of x to a leaf. If at node v on path, x moves left, add all leaves in right subtree of v to our set of solutions.
- Similarly, follow path of x', & whenever x' moves right, we report all points in left subtree.



- If p a leaf, $x \leq p \leq v_{split}$, then p will be reported when paths for x & p diverge. Otherwise, if paths don't diverge, p is reported at the leaf p.
- Sim if $v_{split} < p \leq x'$.

Complexity: - time $O(\log_2 n)$ to follow path of x.
- likewise of x'.
- time required to report k solutions is O(k).
- Total complexity: $O(\log_2 n + k)$.

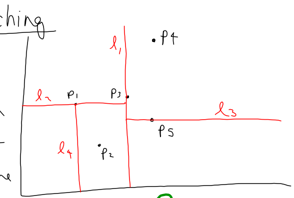
no. of leaves

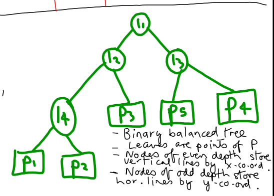
no of solutions

lis 8-16:18

2-d orthogonal range searching

- Set $P \subseteq \mathbb{R}^2$ (assume no 2 have same x or y co-ord.)
- Using vertical line, split through median point ordered by x-co-ordinate. Count point on the line in the left region, should be same number of points in either region or 1 more in the left.
- Now split left & right regions using hor. lines, through pt. with median y-coord, so lower (left) region contains line & & has same no. of points as upper (right) region or 1 more
- Now split using vertical, then horizontal, & keep repeating until each region contains a point.





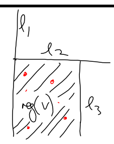
- Binary balanced tree
- Leaves are points of P
- Nodes of even depth store vertical lines by x-co-ord.
- Nodes of odd depth store hor. lines by y-co-ord.

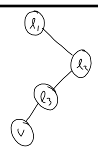
lis 8-16:35

- Called KD-tree.
- Take O(n) storage space, where n number of leaves.
- $O(n \log n)$ to construct KD-tree on n points.

lis 8-16:55

- Region of a node v: a rectangular region bounded by ancestors of v.





- Region(root) = $\mathbb{R}^2$
- Region(lc(l)) = Region(l) ∩ left(l)

  left child
- Region(rc(l)) = Region(l) ∩ right(l).
- A point p of P belong to the subtree under l ⟺ it belongs to region(l).
- Given $[x,x'] \times [y,y']$, find points of P in this rectangle.
- We search through subtree under a node v ⟺ region(v) intersects search rectangle.

  See E-Learning for pseudocode.

lis 8-16:57

Fixing problem that points can't have same x or y-co-ord

Observation: did not need points to be real numbers - only to be elements of a totally ordered set: so we can compute medians & compare elements.

- Pass from $\mathbb{R}$ to $C = (\mathbb{R} \cup \{-\infty, \infty\})^2$ — elements of form $(a|b)$

C has lexicographic order: $(a|b) < (c|d) \iff a < c$ or $(a = c \;\&\; b < d)$

$(p_x, p_y) = p \in P \subseteq \mathbb{R}^2 \longmapsto \hat{p} = ((p_x|p_y), (p_y|p_x)) \in \hat{P} \subseteq C^2$

- No 2 points in $\hat{P}$ have same first or second co-ordinate.

• Let $\hat{R} = [(x|-\infty), (x'|\infty)] \times [(y|-\infty), (y'|\infty)]$

• Then $p \in R \iff \hat{p} \in \hat{R}$.

therefore we only need to run old algorithm on $(\hat{P}, \hat{R})$ instead.

e.g. $(p_x|p_y) \in [(x|-\infty), (x'|\infty)]$
$\iff (x|-\infty) < (p_x|p_y) < (x'|\infty)$

First inequality: $x < p_x$ or $x = p_x \sim x \le p_x$

lis 8-17:09

Second approach - range trees

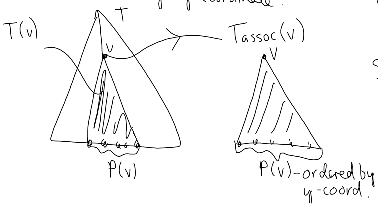
Idea: Given $P \subseteq \mathbb{R}^2$ & $R = [x, x'] \times [y, y']$

1. Use a 1-d search to find points of P whose x-coord belongs to $[x, x']$.
2. Search amongst these points to find those whose y-coord belongs to $[y, y']$.

lis 8-17:25

Data structure: range tree

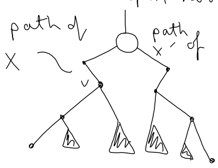
- A binary tree whose leaves are elements of P, ordered by x-coordinate (assume no 2 points have same x or y coord)
- Each node v determine subtree $T(v)$ with set of $P(v)$. For each node we have another binary tree, with leaves $P(v)$, ordered by y-coordinate. $T_{assoc}(v)$.



Storage: $O(n \log n)$
- See E-Learning

lis 8-17:28

- Searching a range tree T for $[x, x'] \times [y, y']$.
- Look at tree ordered by x-coord, find split node for x & x'



- If path for x moves left at v, each leaf in right subtree of v belongs to $[x, x']$
- Hence use a 1-d range search on $T_{assoc}(rc(v))$ to find those whose y-coord belongs to $[y, y']$.
- If v is a leaf, test whether it belongs to R.
- Similarly search path of x'.

Complexity $O(\log^2 n + k)$.

lis 8-17:39