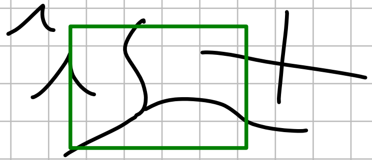
Geometric DS — objects — points in $\mathbb{R}^d$, lines, polylines, polytopes, splines, conic sections

queries:
- exact match
- region:
  - range: d-dim. box $[a_1,b_1]\times\ldots\times[a_d,b_d]$
  - partial match — special case: unbounded intervals $\mathbb{R}$, $[a_d,+\infty)$
  - polygon/polytope

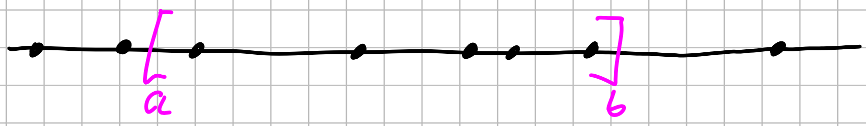
answer:
- enumerate
- one
- count → aggregation



We will consider points in $\mathbb{R}^d$ & (generalized) ranges. & enum/count.

Mostly static.

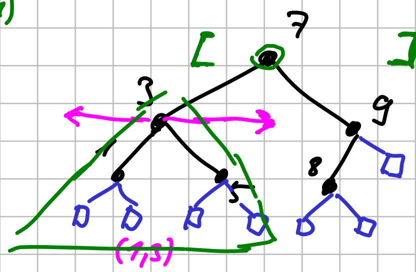
Range queries in 1 dim.



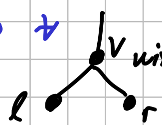
① sort & binary search

Build: $O(n\log n)$ Space: $O(n)$ Query: $O(\log n)$

② binary search trees



Df: For a node v: $\mathrm{int}(v) := \{x \mid \text{searching for } x \text{ visits } v\}$

- $\mathrm{int}(\mathrm{root}) = \mathbb{R}$
- $\forall$  $\mathrm{int}(\ell) = \mathrm{int}(v) \cap (-\infty, k)$, $\mathrm{int}(r) = \mathrm{int}(v) \cap (k, +\infty)$

- by induction $\forall v$ $\mathrm{int}(v)$ is an interval
- keys of all descendants of v lie in $\mathrm{int}(v)$
- also works for external nodes

$S \subseteq \mathbb{R}$ is an interval $\equiv$ $\forall x,y,z \in \mathbb{R}$ $x<y<z$: $x,z \in S \Rightarrow y \in S$

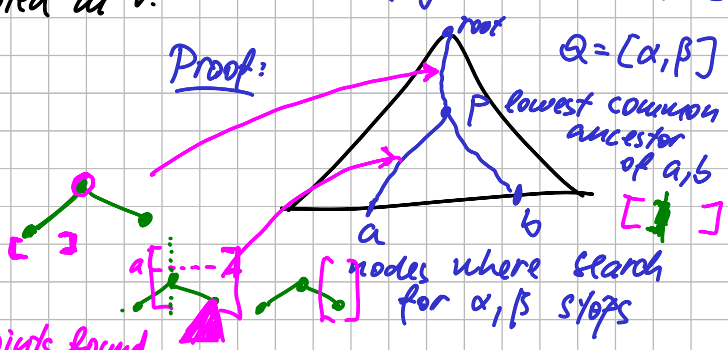
RangeQuery(v, Q): ← root of a subtree; ← query range

1. If v is external, return.
2. If $\mathrm{int}(v) \subseteq Q$: report the subtree rooted at v.
3. If $\mathrm{key}(v) \in Q$: report $\mathrm{key}(v)$
4. $Q_\ell \leftarrow Q \cap \mathrm{int}(\ell(v))$
   $Q_r \leftarrow Q \cap \mathrm{int}(r(v))$
5. If $Q_\ell \neq \emptyset$: RangeQuery($\ell(v)$, Q_ℓ)
6. If $Q_r \neq \emptyset$: RangeQuery($r(v)$, Q_r)

Corollary: R.Q. runs in time $O(\log n + p)$ ← # points found

Lemma: RangeQuery on a balanced BST visits $O(\log n)$ nodes & subtrees.

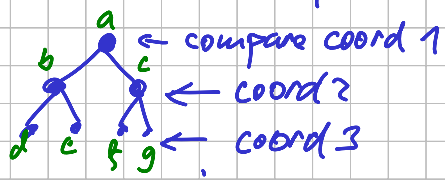
Proof: $Q = [\alpha, \beta]$



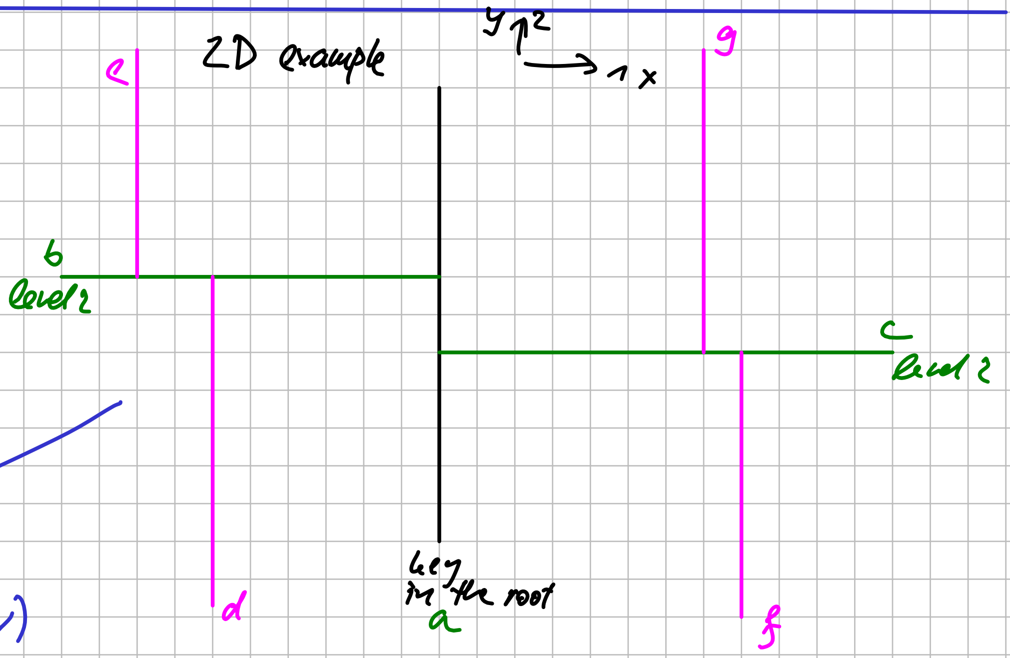
Extensions: ① counting queries: precompute subtree sizes, in step ?. add subtree size to a running total
↳ $O(\log n)$

② can be made dynamic

k-d search trees in $\mathbb{R}^k$



at level i, we compare coord. $i \bmod k$



To every node of the tree we can assign a 2D range $\mathrm{int}(v)$

Temporarily assume: no 2 points share a coordinate.

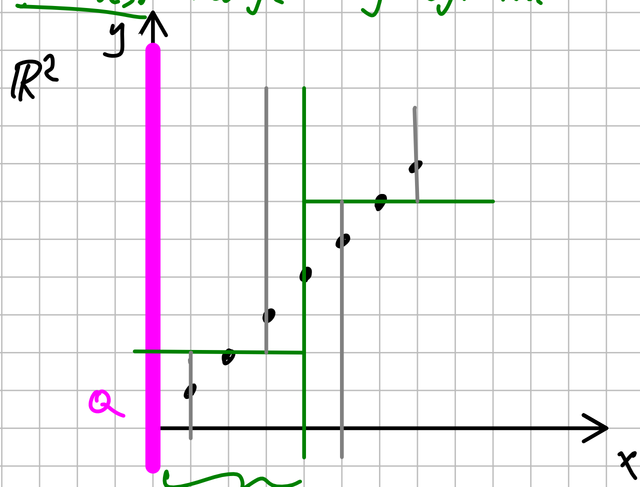
Build: In the root, find m ← median 1st coord.
↳ key in the root (with the corresp. point p_m)
↳ recurse on two half-spaces, but use the next coord.

space is $O(n)$

Build takes $O(n \log n)$ time

height of the tree is $O(\log n)$

Queries: RangeQuery alg. still works, but it's slow:

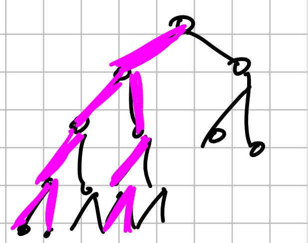


$n := 2^t - 1$

$X := \{(i,i) \mid i \in \{1..n\}\}$

$Q := \{0\} \times \mathbb{R}$

query — x step ⇒ 1 child
— y step → both children



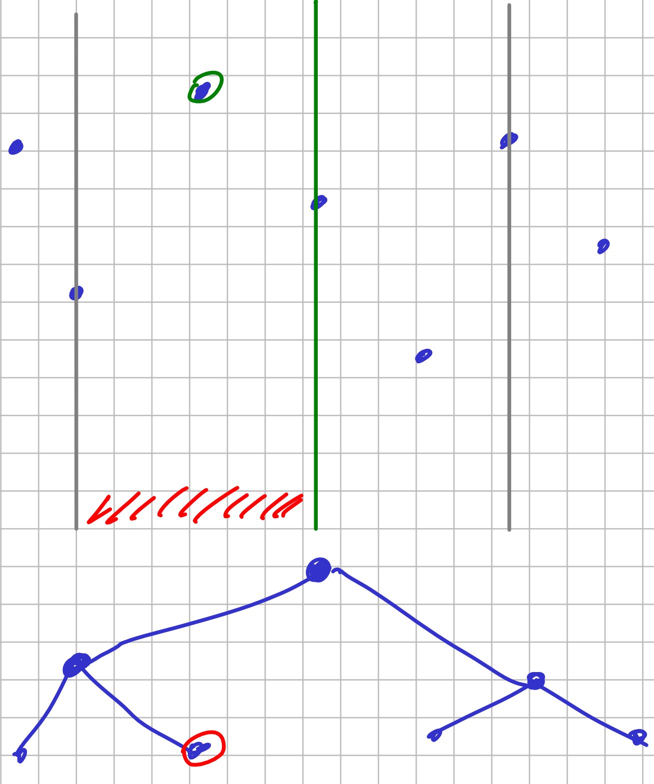
leaves visited $= 2^{\text{height}/2} = 2^{t/2} = \sqrt{2^t} \approx \sqrt{n}$

Query runs in $\Theta(\sqrt{n})$ time!
↳ $\Theta(n^{1-1/d})$ in $\mathbb{R}^d$

... but this is the worst case.

More bad news: ① this is optimal for linear space
② hard to make dynamic

k-dim. Range Trees



① Special case: $k=2$, no shared coords

- x-tree (BST with range queries on the 1st coord)

  ↳ recursive subdivision of $\mathbb{R}^2$ to bands $(a_i, b_i) \times \mathbb{R}$

- y-tree for each band: all points inside the band sorted by y coord

Space: $O(n \log n)$ < $O(n)$ for x-tree; each point is in $O(\log n)$ y-trees

Build: 1. Sort all points by x and by y } both will be passed to recursion } $O(n \log n)$

2. Rec. function:
   - $m \leftarrow$ median x coord → current node of x-tree
   - build y-tree for that x-node
   - recurse on sub-bands

   } one step is $O(\#\text{points}) = O(\text{size of y-tree})$

↳ $O(n \log n)$ time

Query for $[a_1, b_1] \times [a_2, b_2]$:
- ask x-tree for $[a_1, b_1]$ $\xrightarrow{O(\log n)}$ $O(\log n)$ points → check y coord. $O(\log n)$

  ↳ $O(\log n)$ subtrees → bands → ask the y-tree $[a_2, b_2]$ $O(\log^2 n)$ $+p$

↳ $O(\log^2 n)$ time $+p$

② Shared x coords: for every node of the x-tree add a second y-tree for points matching exactly the x coord.

↳ only const-times slower

③ More dimensions:
- primary tree on 1st coord → subdiv. space to k-dim. bands
- for each band: secondary tree — $(d-1)$-dim. range tree

By induction:
Space $O(n \cdot \log^{d-1} n)$
Build $O(n \cdot \log^{d-1} n)$
Query $O(\log^d n + p)$

→ can be made dynamic (as exercise)

↳ can improve query to $O(\log^{d-1} n)$ (ex.)