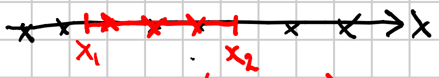
Orthogonal Range Searching

Notetitel 22-08-2008

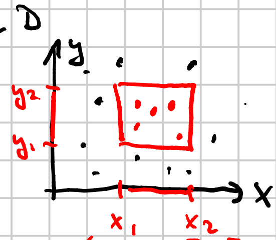
Problem Preproces a set of n d-dimensional points, to support (axis aligned) d-dimensional rectangle queries.

1D



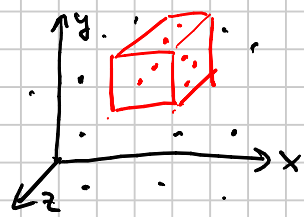
Query($[x_1, x_2]$)

2D



Query($[x_1, x_2] \times [y_1, y_2]$)

3D



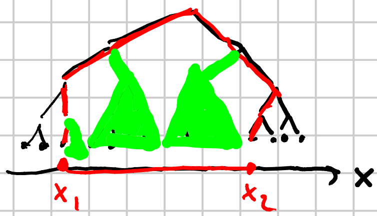
Query($[x_1, x_2] \times [y_1, y_2] \times [z_1, z_2]$)

Variations

- Preprocessing time
- Query time } Trade-off
- Space
- Dimension
- Static vs Dynamic point set
- Comparison model vs Integer coordinats
- Other queries: Count #points in region / Return point with max associated value / Return sum of points associated values

1D

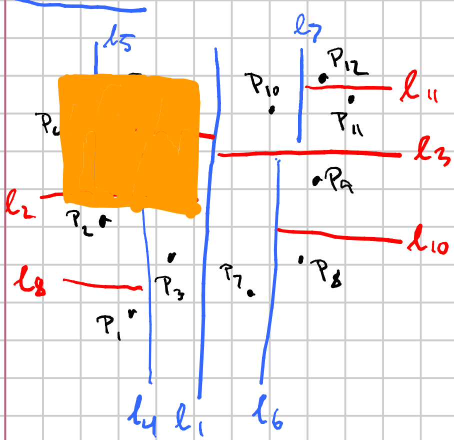
Store points in search tree (elements at the leaves)

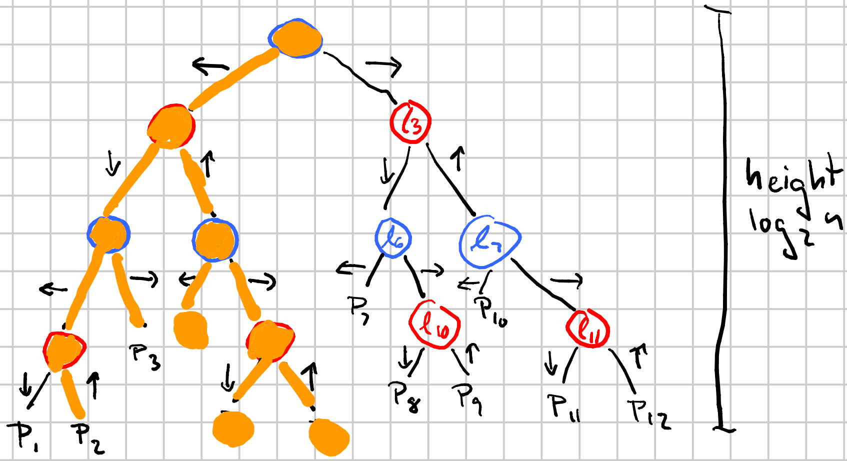


Report all subtrees between paths to x_1 and x_2

- Preprocessing $O(n \cdot \log n)$
- Space $O(n)$
- Query $O(\log n + k)$ ← output size

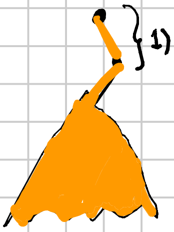
kd-tree





- Space $O(n)$
- Preprocessing $O(n \cdot \log n)$ — each element participates in one selection per level
- Query $O(\sqrt{n} + k)$ — top down traverse all nodes intersecting query rectangle

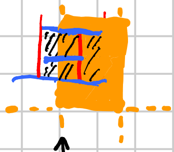
kd-tree – Query Analysis



Nodes visited:

1) Node's rectangle completely contains query
   $\Rightarrow$ At most $\log_2 n$ nodes
2) Node's rectangle contained in query rectangle
   $\Rightarrow$ Complete subtree reported, i.e. charged to k since k_i leaves reported and k_i internal nodes

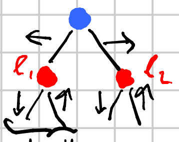
3) Node partially overlaps with query (shaded area)
   $\Rightarrow$ Node or a child stores a separating segment that is intersected by one of the 4 (infinite) lines defining the query rectangle.



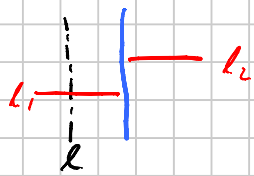
side intersected

Fact: Any horizontal/vertical line can at most be intersected by $O(\sqrt{n})$ nodes.

$$I(n) \leq 1 + 2\,I\left(\frac{n}{4}\right)$$
$$I(n) = O\left(2^{\log_4 n}\right) = O(n^{1/2})$$



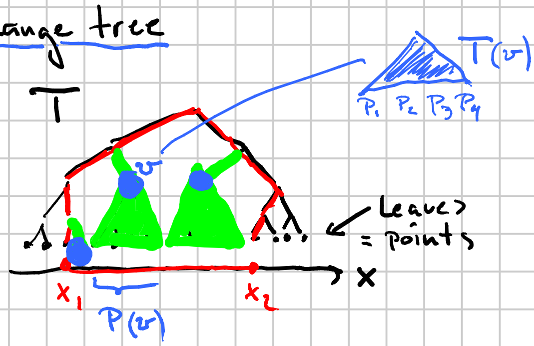
Only these can intersect ℓ



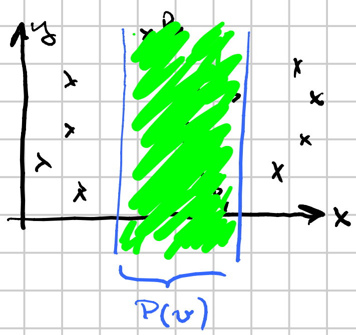
Total #nodes visited: $O(\sqrt{n} + k)$

Note: kd-trees can also support other (non-orthogonal) query-shapes, by recursive traversal of nodes intersecting query range / bounding box of query range

Range tree



- Nodes where subtree contains point within x-range
- $\leq 2 \log n$ subtrees with points within range
- Store each set $P(v)$ as a search tree T_v sorted w.r.t. y-coordinate



Queries: $O(\log^2 n + k)$ – search in $\leq 2 \log n$ $T(v)$ trees

Space: $O(n \cdot \log n)$ – each point stored in $\log n$ $T(v)$ trees at ancestors in T

Preprocessing: $O(n \cdot \log n)$ – construct $T(v)$ lists by merging childrens $T(v)$ lists

Higher dimensions

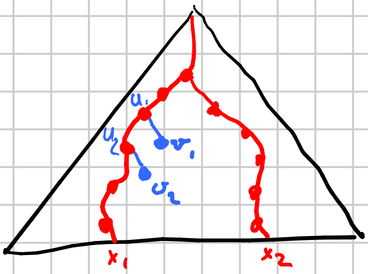
kd-trees: Space $O(n)$, Query $O(n^{1-1/d}+k)$, Preprocessing $O(n\cdot\log n)$

- Round-Robin split w.r.t. the d-levels
- Only every d'th level is parallel wrt to a side in query, i.e. only one child can contribute to the output

Range trees: Space $O(n\cdot\log^{d-1} n)$, Query $O(\log^d n+k)$, Preprocessing $O(n\cdot\log^d n)$

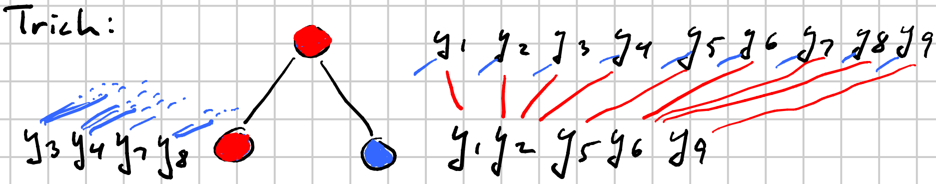
- Build T on one dimension – each $T(v)$ structure is a range tree for d-1 dimensions
- Query : $T(d,n) \le 2\log n\cdot T(d-1,n)$, $T(1,n)=\log n$
  $\Rightarrow T(d,n) = O(2^d\cdot\log^d n)$
- Space : $S(d,n) \le O(\log n)\cdot S(d-1,n)$, $S(1,n)=O(1)$
  $\Rightarrow S(d,n) = O(\log^{d-1} n)$

Fractional cascading



Goal : Search in $T(v_1)$ and $T(v_2)$ for y_1
- but avoid using $O(\log n)$ time at each node

Trick:



- Add links from each point in $T(v)$ to its immediate predecessor/successor w.r.t. y-value in both child lists
- Only need to search for y_1 at root in $O(\log n)$ time.

Space $O(n\cdot\log n)$, Query $O(\log n+k)$, Preprocessing $O(n\cdot\log n)$

Summary of Results (2D)

	Preprocessing	Space	Query
kd-trees	$O(n\cdot\log n)$	$O(n)$	$O(\sqrt{n}+k)$ *
Range trees	$O(n\cdot\log n)$	$O(n\cdot\log n)$	$O(\log^2 n+k)$
-"- + fractional cascading	$O(n\cdot\log n)$	$O(n\cdot\log n)$	$O(\log n+k)$
Chazelle	$O(n\cdot\log n)$	$O\left(\frac{n\cdot\log n}{\log\log n}\right)$	$O(\log n+k)$ **

* Optimal query for $O(n)$ space

** Optimal space for fastest possible queries.