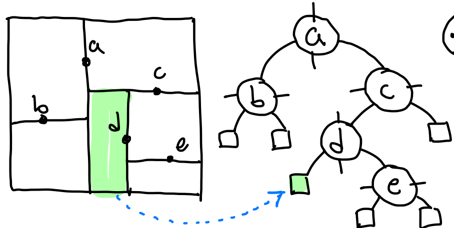
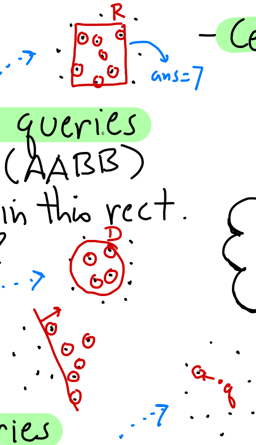
Kd-Tree Queries I



Queries?

- Orthogonal range queries
  - Given query rect. (AABB) count/report pts in this rect.
- Other range queries?
  - Circular disks
  - Halfplane
  - ⋮



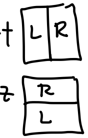
- Nearest neighbor queries
  - Given query pt, return closest pt in the set
  - Find k^{th} closest point
  - Find farthest point from q

This Lecture: $O(\sqrt{n})$ time alg for orthog. range counting queries in $\mathbb{R}^2$

$\longrightarrow$ General $\mathbb{R}^d$: $O(n^{1-1/d})$

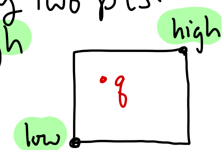
Kd-Trees:

- Partition trees
- Orthogonal split → vert / horz
- Alternate cutting dimension x, y, x, y, ...
- Cells are axis-aligned rectangles (AABB)



Axis-Aligned Rect in $\mathbb{R}^d$

- Defined by two pts: low, high

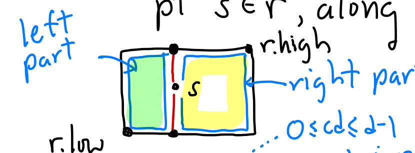


- Contains pt $q \in \mathbb{R}^d$ iff

$$low_i \le q_i \le high_i \quad 1 \le i \le d$$

Rectangle methods for kd-cells:

- Split a cell r by a split pt $s \in r$, along cutdim cd



$0 \le cd \le d-1$

pt in r

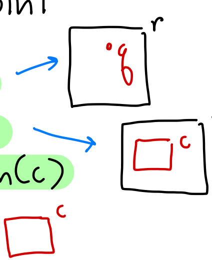
r.leftPart(cd, s)
→ returns rect with low = r.low + high = r.high but high[cd] ← s[cd]

r.rightPart(cd, s)
→ high = r.high + low = r.low but low[cd] ← s[cd]

Useful methods:

Let r, c - Rectangle
q - Point

- r.contains(q)
- r.contains(c)
- r.isDisjointFrom(c)

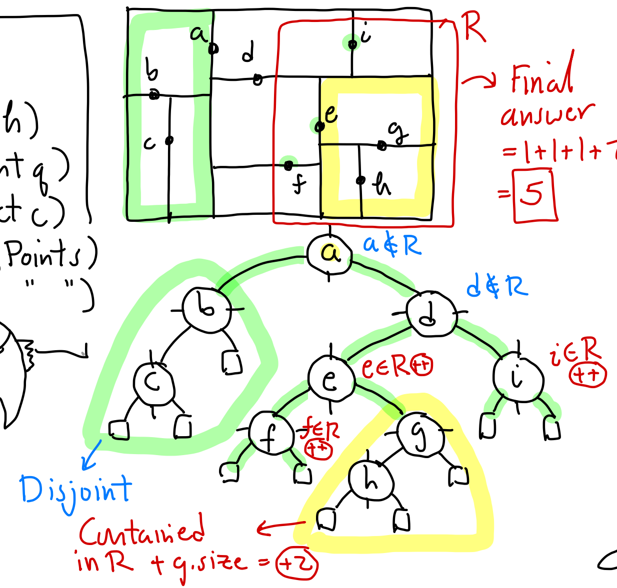


Kd-Tree Queries II

Orthog. Range Query

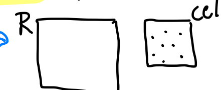
- Assume: Each node p stores:
  - p.pt : splitting point
  - p.cutDim : cutting dim
  - p.size : no. of pts in p's subtree
- Tree stores ptr. to root and bounding box for all pts.
- Recursive helper stores current node p + p's cell.

```
class Rectangle {
    private Point low, high
    public Rect(Point l, Point h)
      "     boolean contains(Point q)
      "     boolean contains(Rect c)
      "     Rect leftPart(int cd, Point s)
      "     Rect rightPart( "   "   "   " )
}
```

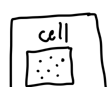


Cases:

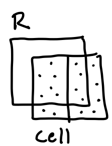
- p == null → fell out of tree → 0
- Query rect is disjoint from p's cell
  → return 0
  → no point of p contributes to answer



- Query rect contains p's cell
  → return p.size
  → every point of p's subtree contributes to answer.



- Otherwise: Rect. + cell overlap – Recurse on both children



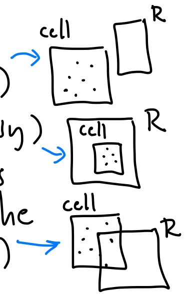

```
int rangeCount(Rect R, KDNode p, Rect cell)
    if (p == null) return 0  // fell out of tree
    else if (R.isDijointFrom(cell)) return 0  // no overlap
    else if (R.contains(cell)) return p.size  // take all
    else { int ct = 0
        if (R.contains(p.pt)) ct++  // p's pt in range
        ct += rangeCount(R, p.left,
                     cell.leftPart(p.cutDim, p.pt))
        ct += rangeCount(R, p.right, cell.rightPart...
    }
```

Kd-Tree Queries III

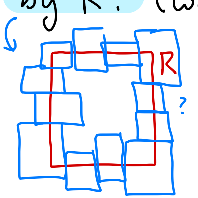
Theorem: Given a balanced kd-tree storing n pts in $\mathbb{R}^2$ (using alternating cut dim), orthog. range queries can be answered in $O(\sqrt{n})$ time.

→ Slower than $\log n$. Faster than n

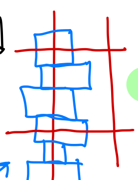
Stabbing: 3 cases
- cell is disjoint (easy)
- cell is contained (easy)
- cell partially overlaps or is stabbed by the query range (hard!)



How many cells are stabbed by R? (worst case)



Simpler: Extend R's sides to 4 lines + analyze each one.



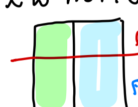
Analysis: How efficient is our algorithm?

→ Tricky to analyze
→ At some nodes we recurse on both children ⇒ $O(n)$ time?
→ At some we don't recurse at all!

Lemma: Given a kd-tree (as in Thm above) and horiz. or vert. line ℓ, at most $O(\sqrt{n})$ cells can be stabbed by ℓ

Proof: w.l.o.g. ℓ is horiz.

Cases: p splits vertically



stab both

Solving the Recurrence:
- **Macho:** Expand it
- **Wimpy:** Master Thm (CLRS)

Master Thm:

$$T(n) = aT(n/b) + n^d \quad \& \quad d < \log_b a$$

$$\Rightarrow T(n) = n^{\log_b a}$$

For us: $a = 2$, $b = 4$, $d = 0$ $\Rightarrow T(n) = n^{\log_4 2} = n^{1/2} = \sqrt{n}$

Since tree is balanced a child has half the pts & grandchild has quarter.

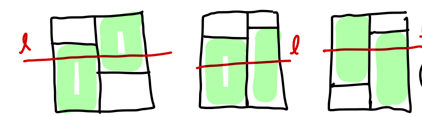
Recurrence: $T(n) = 2 + 2T(n/4)$

2 cells stabbed

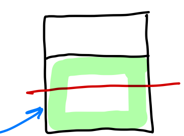
Recurse on 2 grandchildren

Each has $n/4$ pts

If we consider 2 consecutive levels of kd-tree, ℓ stabs at most 2 of 4 cells:



p splits horizontally



ℓ stabs only one