**Kd-Trees:**
- Partition trees
- Orthogonal split
- Alternate cutting dimension $x,y,x,y,...$
- Cells are axis-aligned rectangles (AABB)

**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 $q$

**Kd-Tree Queries**
- Axis-Aligned Rect in $\mathbb{R}^d$
  - Defined by two pts: $low, high$
  - Contains pt $q \in \mathbb{R}^d$ iff
    - $low_i \leq q_i \leq high_i$

**Useful methods:**
- Let $r,c - \text{Rectanglu}$
- $q - \text{Point}$
- $r \cdot \text{contains}(q)$
- $r \cdot \text{contains}(c)$
- $r \cdot \text{isDisjointFrom}(c)$

**Rectangular methods for kd-cells:**
- Split a cell $r$ by a split pt $s \in r$, along cutdim $cd$
  - $r \cdot \text{leftPart}(cd,s)$
    - Returns rect with $low = r \cdot low$
    - $high = r \cdot high \text{ but } high[cd] \leftarrow s[cd]$
  - $r \cdot \text{rightPart}(cd,s)$
    - $high = r \cdot high + low = r \cdot low \text{ but } low[cd] \leftarrow s[cd]$

**This Lecture:** $O(n \log n)$ time alg.

for orthog range counting queries in $\mathbb{R}^2$

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

Cases:
- $p == null$ → fell out of tree → 0
- Query rect is disjoint from $p$'s cell → $R$
  - 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

Kd-Tree Queries II

```
class Rectangle2D {
    private Point low, high
    public Rect (Point l, Point h) {
        boolean contains(Point q) {
            boolean contains(Rect c) {
                Rect leftPart (int cd, Points)
                Rect rightPart ("", ", ")
            }
        }
    }
} 
```

int rangeCount(Rect $R$, KDNode $p$, Rect $cell$) {
    if ($p == null$) return 0 // fell out of tree
    else if ($R$ is Disjoint From ($cell$)) return 0 // overlap
    else if ($R$.contains($cell$)) return $p.size$ // take all
    else { int $ct = 0$ // partial overlap
        if ($R$.contains($p.pt$)) $ct++$ // $p$ pt in range
        $ct += rangeCount(R, p.left, cell.leftPart(p.cutDim, p.pt))$
        $ct += rangeCount(R, p.right, cell.rightPart...)
    }
    return $ct$
}
Announcements 3/28
- Midterm: Almost done
- Prog Assign 2
  Part 1 (20%) - due Wed Apr 5
  Part 2 (80%) - due Wed Apr 19
Programming Assignment 2: Sliding Midpoint kd-Tree

- Extended binary tree
  - Internal nodes: split but no data
  - External nodes: data but no splitting

- Sliding midpoint splitting rule
  - Splitting rule for squares

- But: Two exceptions
  - If all pts are equal along cut dimension: flip to other
  - If all pts on same side of midpoint: slide splitting plane to closest point

- Rebalancing
  - Whenever the number of insertions grows too large (~ half size of subtree)
    rebuild the entire subtree from scratch.
  - How?
    - Traverse subtree + put points in an ArrayList
    - BulkCreate Build tree recursively by applying sliding midpoint
insert: DFW, DCA, LAX, SEA