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 \( q \)

This Lecture: \( O(\sqrt{n}) \) time alg. for orthog. range counting queries in \( \mathbb{R}^2 \)
- General \( \mathbb{R}^d \): \( O(n^{1-1/d}) \)

Kd-Tree Queries

Rectangular methods for kd-cells:
- Split a cell \( r \) by a split pt \( s \in r \), along cutdim \( c \):
  - Orthogonal split \( c \perp \text{horz} \) \( \text{vert} \)
  - Alternate cutting \( c \)

Axis-Aligned Rect in \( \mathbb{R}^d \)
- Defined by two pts:
  - \( \text{low, high} \)
- Contains pt \( q \in \mathbb{R}^d \) iff
  - \( \text{low} \leq q_i \leq \text{high} \)

Useful methods:
- Let \( r, c \) - Rectangles 2D
- \( q \) - Point
- \( r \).contains(\( q \))
- \( r \).contains(c)
- \( r \).isDisjointFrom(c)
Orthogonal 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.

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

Kd-Tree Queries

class Rectangle {
  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 (" " " " " )"
}

void expand (Point q)

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
    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...}

Note: $ct$ will be the final count of points in the range $R$.
Announcements: 11/01

- Midterm 1 Grades - Published soon
- Updated Proj 2 Skeleton
- Homework 3 - Soon