Geometric Search:
- Nearest neighbors
- Range searching

So far: 1-dimensional keys
- Multi-dimensional data
- Applications:
  - Spatial databases + maps
  - Robotics & Auton. Systems
  - Vision/Graphics/Games
  - Machine Learning

Partition Trees:
- Tree structure based on hierarchical space partition
- Each node is associated w. a region - cell
- Each internal node stores a splitter - subdivides the cell

Multi-Dim vs. 1-dim Search?

Similarities:
- Tree structure
- Balance $O(\log n)$
- Internal nodes - split
- External nodes - data

Differences:
- No (natural) total order
- Need other ways to discriminate + separate
- Tree rotation may not be meaningful

Multi-Dimensional data - Tree structure based on hierarchical space partition
- Each node is associated w. a region - cell
- Each internal node stores a splitter - subdivides the cell

Quadtrees & kd-Trees

Representations:
- Scalars: Real numbers for coordinates, etc.
- Points: $p = (p_1, \ldots, p_d)$ in real $d$-dim space $\mathbb{R}^d$
- Other geometric objects: Built from these

Class Point:

```java
float[] coord // coords
Point(int d)
    ... coord = new float[d]
int getDim() -> coord.length
float get(int i) -> coord[i]
... others: equality, distance, toString...
```
Point Quadtree:
- Each internal node stores a point
- Cell is split by horizontal + vertical lines through point

Quadtree: (abstractly)
- Partition trees
- Cell: Axis-parallel rectangle
- Splitter: Subdivides each cell into four (generally 2d) subcells

Find/ Pt Location:
Given a query point q, is it in tree, and if not which leaf cell contains it?
- Follow path from root down (generalizing BST find)

Quadtree & Kd-Trees II

History:
- Bentley 1975
- Called it 2-d tree \((\mathbb{R}^2)\)
- 3-d tree \((\mathbb{R}^3)\)
- In short \(Kd\)-tree (any dim)
- Where/ which direction to split? \(\rightarrow\) next

Kd-Tree: Binary variant of quadtree
- Splitter: Horizontal or vertical line in \(2-d\) (orthogonal plane \(aw\))
- Cell: Still AABB
  - left: left/below
  - right: right/above

Each external node corresponds to cell of final subdivision
Example:

Kd-Tree Node:

```java
class KDNode {
    Point pt // splitting point
    int cutDim // cutting coordinate
    KDNode left // low side
    KDNode right // high side
}
```

Quadtrees & Kd-Trees III

Analysis:
Find runs in time $O(h)$, where $h$ is height of tree.

Theorem:
If pts are inserted in random order, expected height is $O(\log n)$

Value
```java
public boolean onLeft(Point q)
    {return q[cutDim] < pt[cutDim]}
}
```
KD-Tree Insertion:

```
KdTreeNode insert(Point x, Value v, KdTreeNode p, int cd){
    if (p == null) // fell out?
        p = new KdTreeNode(x, v, cd) // new leaf node
    else if (p.pt == x)
        Error! Duplicate key
    else if (p.onLeft(x))
        p.left = insert(x, v, p.left, (cd+1) % dim)
    else
        p.right = insert(x, v, p.right, (cd+1) % dim)
    return p
}
```

Deletion:

- Descend path to leaf
- If found:
  - leaf node → just remove
  - internal node → find replacement
    - copy here
    - recur. delete replacement

Rebalance by Rebuilding:
- Rebuild subtrees as with scapegoat trees
- O(log n) amortized
- Find: O(log n) guaranteed.

Example:
```
insert(3,4)
```

Analysis:
Run time: \(O(h)\)

Can we balance the tree?

- Rotation does not make sense

Tree height