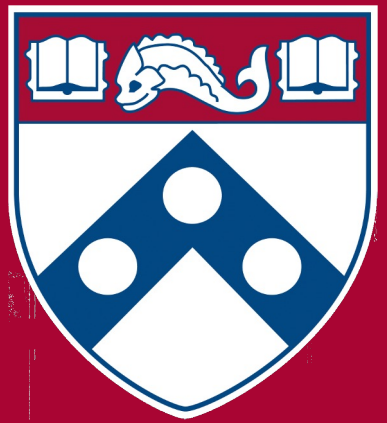
QuadTrees

CIT5940



Multidimensional Keys

- Key composed of more than one attribute
- In many applications, records need to be searched based on more than one attribute.
- BST is efficient for searching on a one-dimensional key (range query)
- BST does not work well for multidimensional range query
    - How to find all elements between (1, 1) and (4, 4)??

Multidimensional Range Query

- Find all restaurants within two miles of my house
- Data need to be searched using two-dimensional key (*x* and *y* coordinates)
- All dimensions have the same importance

Spatial DS

- Spatial Data:
  - A location identified by *x* and *y* coordinates
- Spatial DS allow efficient processing of multidimensional range queries

PR QuadTree

- Point-Region quadtree
- Invented by Rafael Finkel and Jon Louis Bentley
- **Full four-way branching (4-ary) tree**

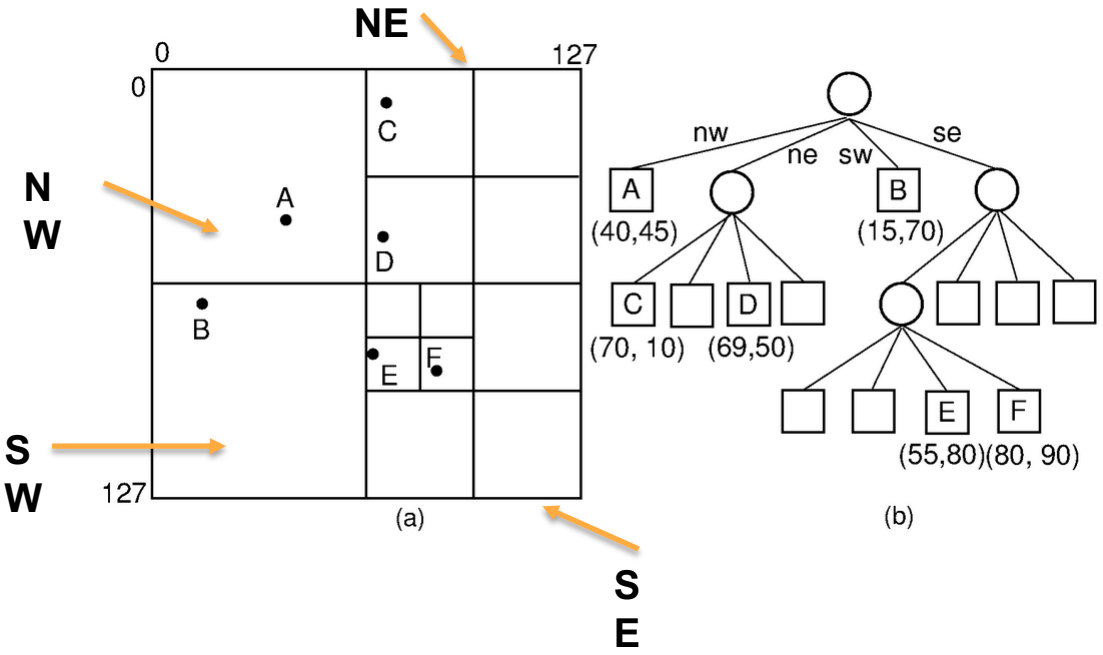
PR QuadTree

- Each node either has exactly four children or is a leaf
- Represents a collection of data points in two dimensions
- Each region is represented by a Node / quadrant

PR QuadTree

- Internal nodes do not store data (points)
- A leaf node represents a region containing zero or one data point
- A region containing more than one point is repeatedly subdivided into 4 equals subquadrants until no leaf node contains more than a single point

PR QuadTree



- Example: Google Maps Utility Library https://developers.google.com/maps/documentation/ios-sdk/utility/quadtree

PR QuadTree: Insert

```
INSERT(Node node, Point p)
1. if (node.isNotEmpty())
2.    SPLIT(node)
3.    nextNode = CHOOSE_LEAF(node, p)
4.    INSERT(nextNode, p)
5. else if (node.isLeaf())
6.    node.setKey(p)
7. else
8.    nextNode = CHOOSE_LEAF(node, p)
9.    INSERT(nextNode, p)
```

```
CHOOSE_LEAF(Node node, Point p)
1. Let x be the child of node whose
   region contains p.
2. return x
```

```
SPLIT(Node node)
1. Split the region occupied by node into four
   equally-sized regions SW, NW, SE and NE.
2. Create four children for node and assign them to
   occupy the quadrants SW, NW, SE and NE in that order.
3. p = node.getKey()
4. node.deleteKey()
5. leaf = CHOOSE_LEAF(node, p)
6. leaf.setKey(p)
```

Breadth-first Search (BFS)

- Visits nodes/regions one level at a time (from root to leaves)
- Siblings of a node are visited before its children

BFS Algorithm

```
Initialize empty queue Q
ENQUEUE(Q, root)
while Q not empty do
    u ← DEQUEUE(Q)
    for each v ∈ Children(u) do
        ENQUEUE(Q, v)
```

Example

- HW4 – Blocky