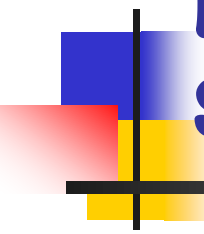


Data Structures:
Spatial Trees: Quad Tree

YoungWoon Cha
(Slide credits to Won Kim)
Spring 2022

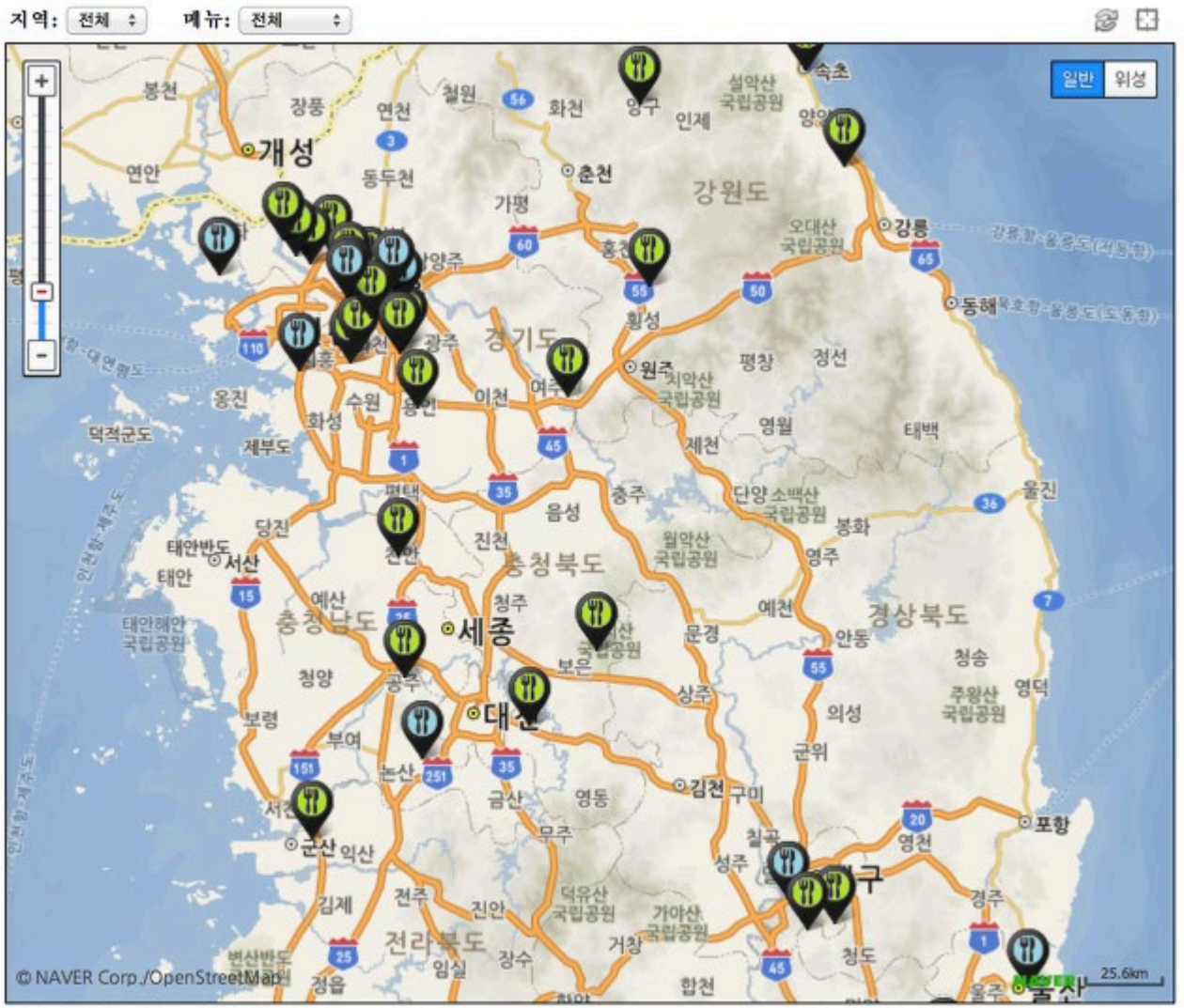
Characterizing Trees

Tree Properties

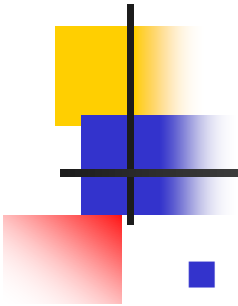
- Degrees of Trees
  - unary tree
    - linked list
  - binary tree
    - binary search tree
  - m-way tree (m>2)
- Height Balanced
  - unbalanced
  - balanced (but not perfectly)
    - AVL tree, T-tree
  - perfectly balanced
    - 2-3 tree
- Dimension
  - one dimension
    - AVL tree, T-tree
  - n-dimension (n >=2)
    - quad tree, kd tree
- Interior node has no value
  - trie, region quadtree

Spatial Data Structures

Storing and Searching for Popular Restaurants on a Map



* 자료출처: 이영돈PD의 먹거리X파일
* 사용법: 표시클릭 - 상세정보 보기, 더블클릭 - 확대.
* 데이터 날짜: 2013.9.27.
* 댓글 남기기: 바로가기



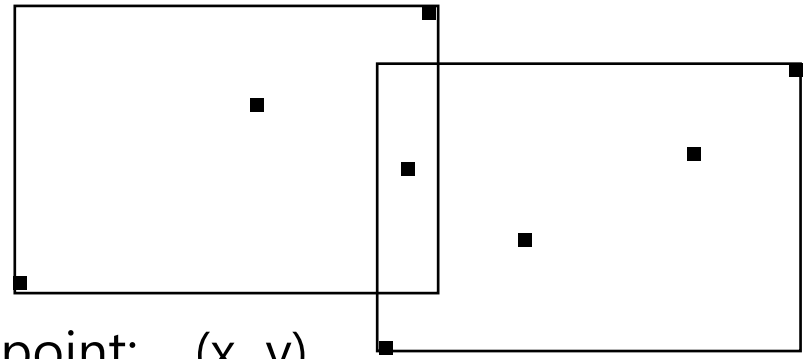
Spatial Data Structures

- QuadTree, OctTree
- k-d Tree, Grid File

Spatial vs. Non-Spatial Data Structures

- Non-Spatial Data Structures
  - Indexing 1-dimensional data
  - Key value is a single data item
    - e.g., 250, "Hong Gil Dong",…
- Spatial Data Structures
  - Indexing multi-dimensional data
  - Key value is a set of data items
    - E.g., (75, 127), (15, 46, 93), ("Hong Gil Dong", 25, thief, "running on the roof")

Point and Bounding Rectangle

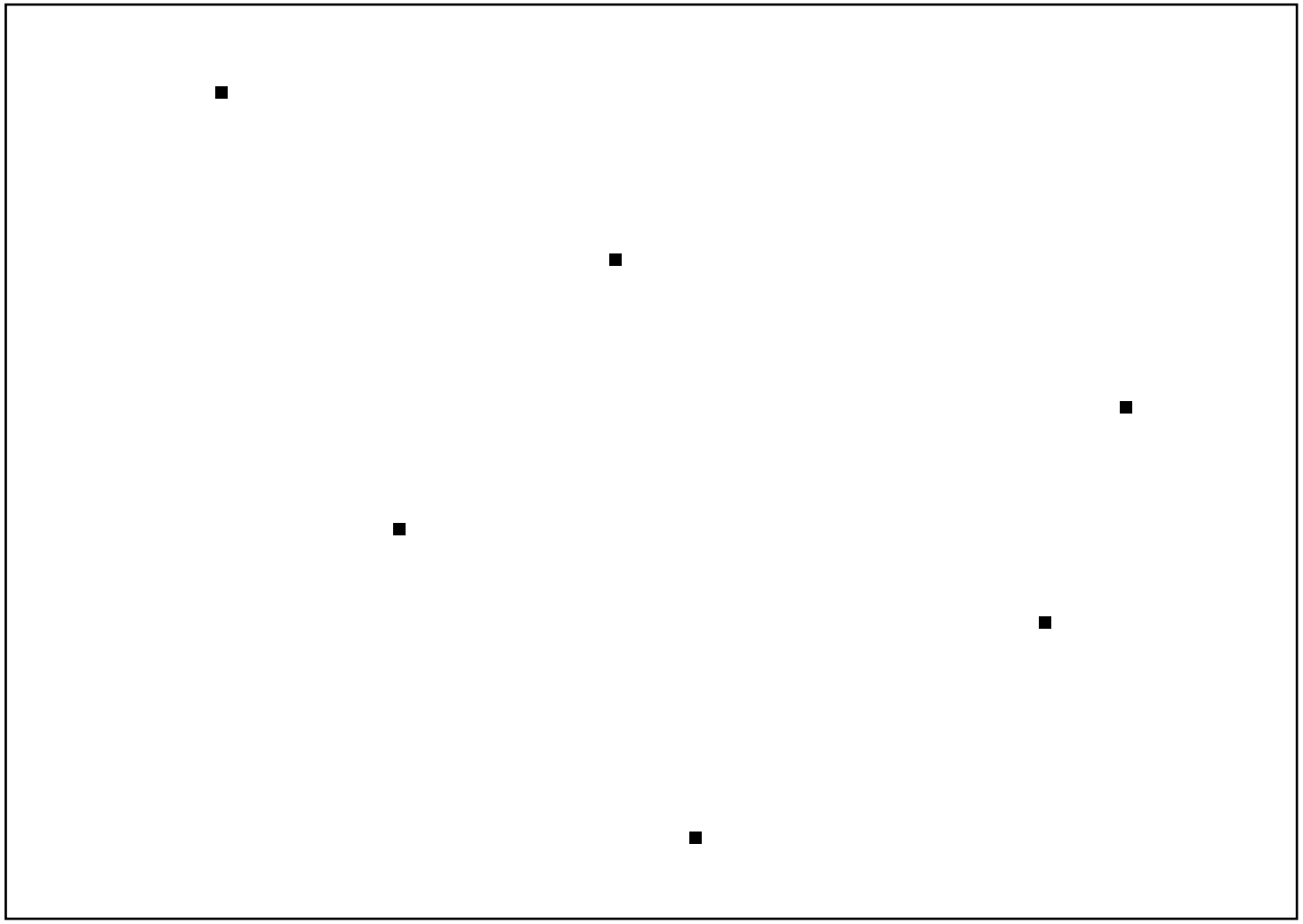


- point: (x, y)
- bounding rectangle (BR): ((x1,y1), (x2,y2))
- Relationship between a BR and a point
  - A point is contained in the BR
  - A point lies on an edge of the BR
  - Distance of the point from the center of the BR
- Relationship between two BRs
  - One BR contains the other BR
  - The two BRs touch each other
  - Distance between the BRs

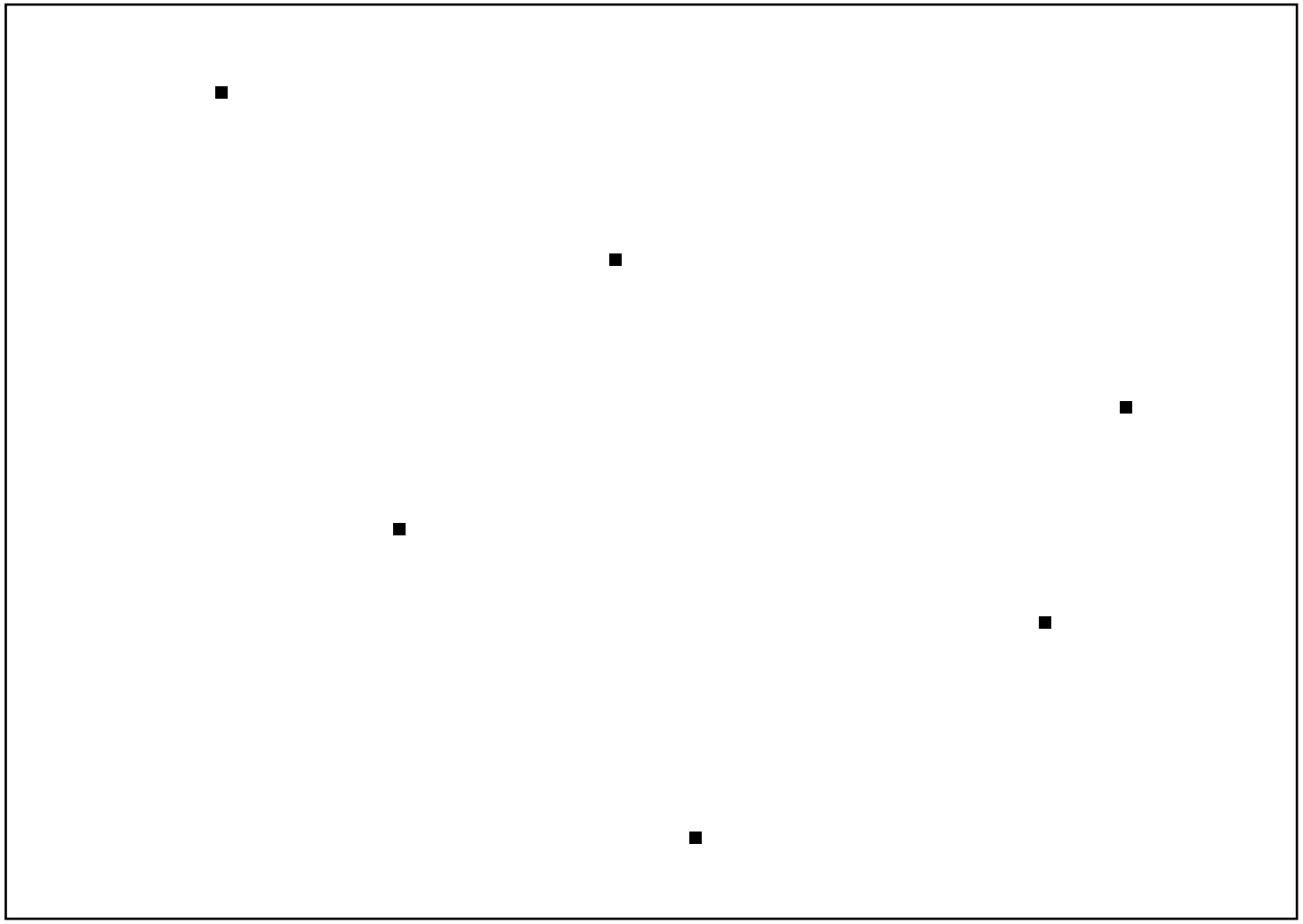
(Region) QuadTree

- Tree of Degree 4
- Performance: $O(\log_4 n)$
- Not Balanced
- Divides a 2-Dimensional Space into 4 Quadrants (Regions)
- Reading
  - http://www.cs.ubc.ca/~pcarbo/cs251/welcome.html

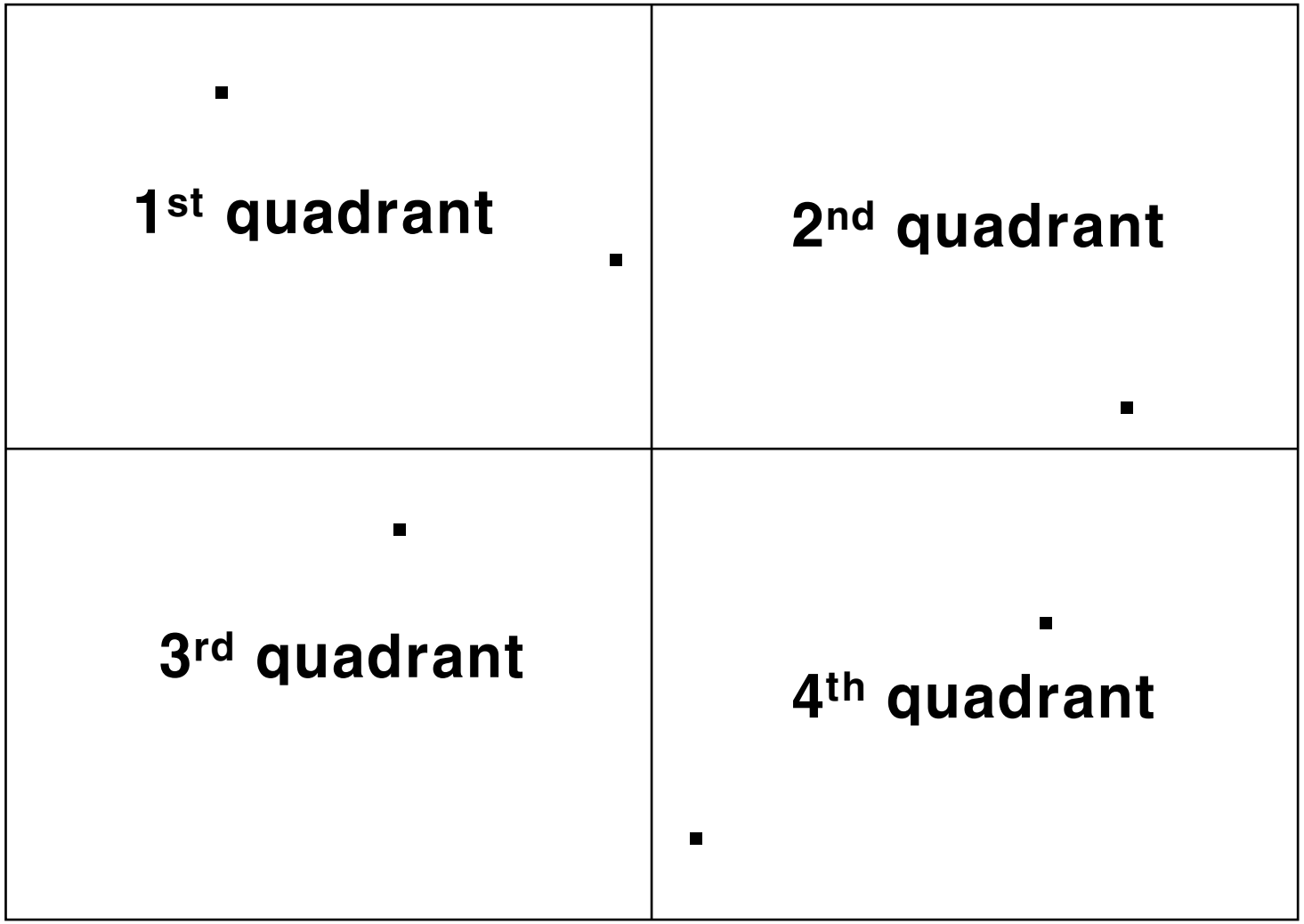
Example: Find Restaurants on the Map of a City (Each Point Represents All Data About a Restaurant.)



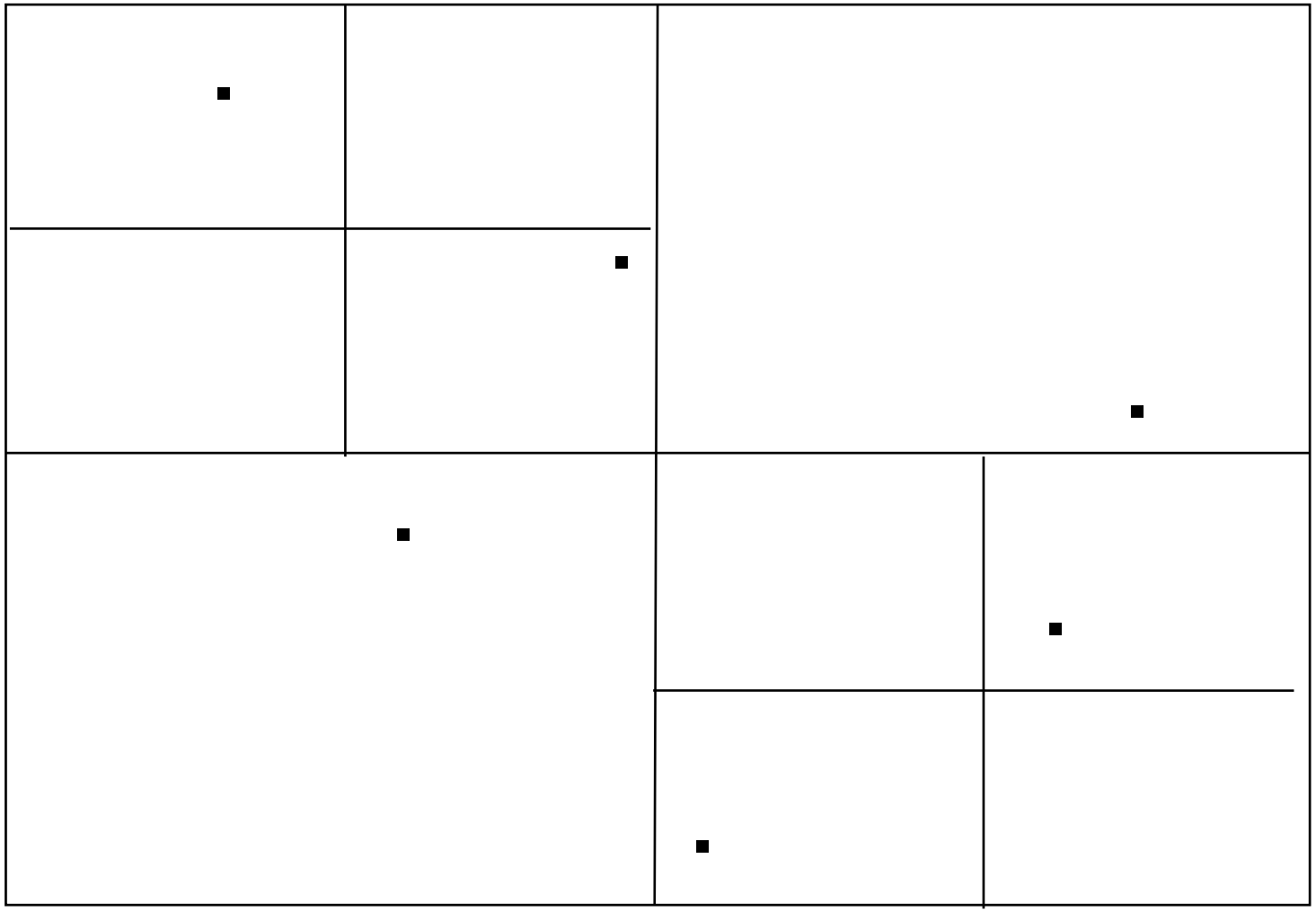
Divide a 2-D Space into Successive Quadrants



Divide a 2-D Space into Successive Quadrants

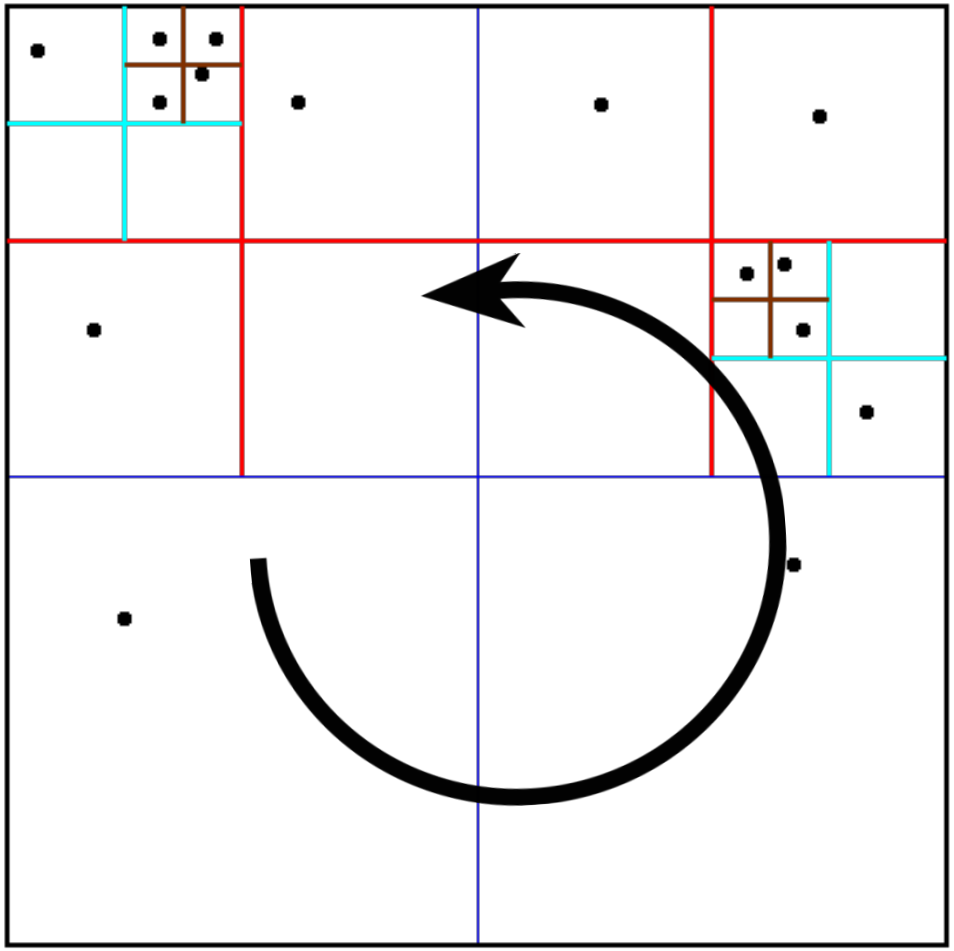


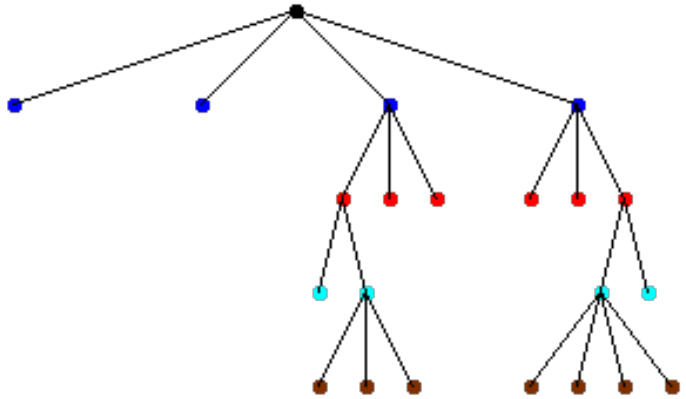
Divide a 2-D Space into Successive Quadrants (until each quadrant has only one point in it)



Map the Quadrants to Nodes of a QuadTree:
specify the starting quadrant and direction of mapping

Adaptive quadtree where no square contains more than 1 particle





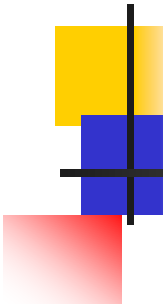
〈counter-clockwise
from the 3rd quadrant〉

Region QuadTree Nodes

- Region quadtree is a type of trie (to learn later).
- Interior nodes contain pointers to child nodes, but no data.
- Data are stored on the leaf nodes.

Leaf Node

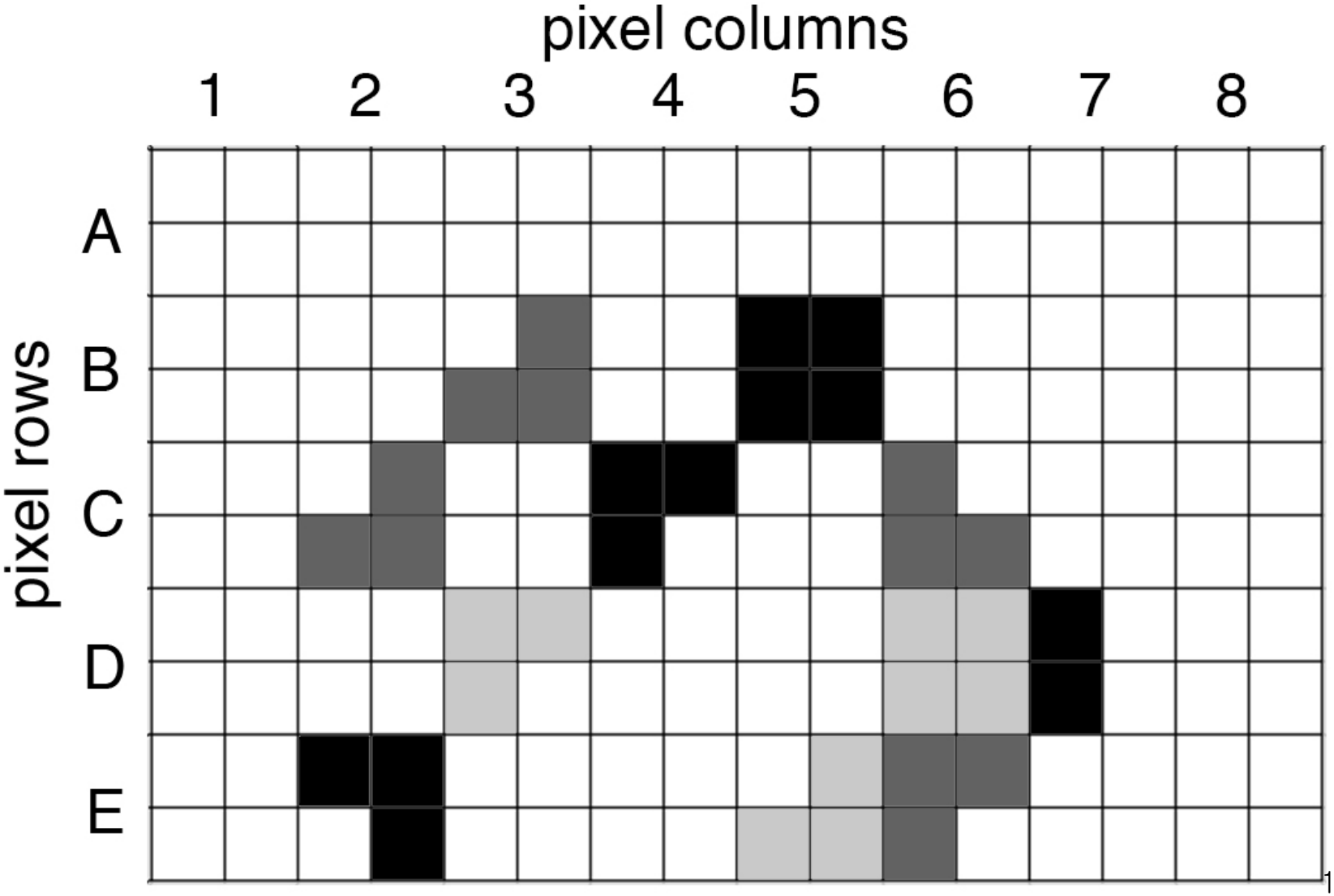
- Each leaf node contains data corresponding to a specific subregion, for example,
  - point, number, string, pointer to a file,…
  - line, shape, pixels, image elements,…
  - the latitude and longitude of a set of cities
  - the average temperature over the subregion it represents



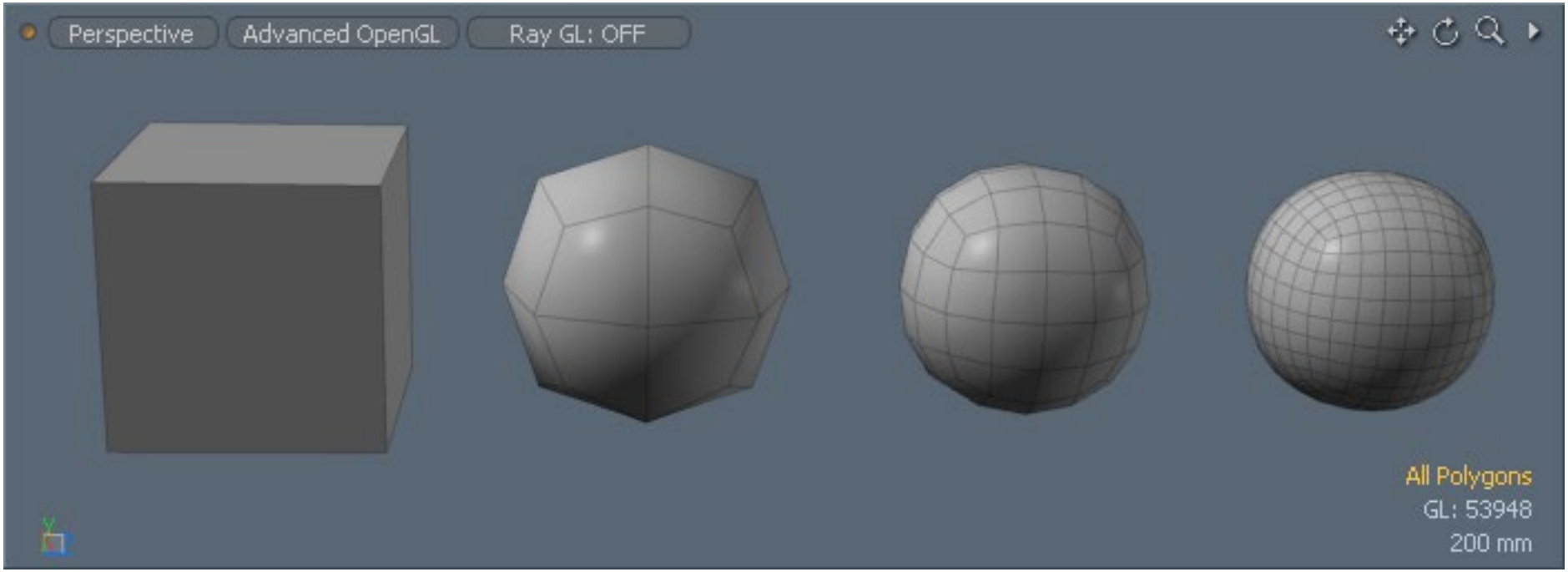
Applications

- software that searches for segments of stored color images
- software that searches for information about restaurants (schools, churches, car centers,…) within a region of a city (province, country)
- software that searches for parts of geometry objects
- supporting collision detection (intersection of two solid objects) in video games, physical simulations, computational geometry

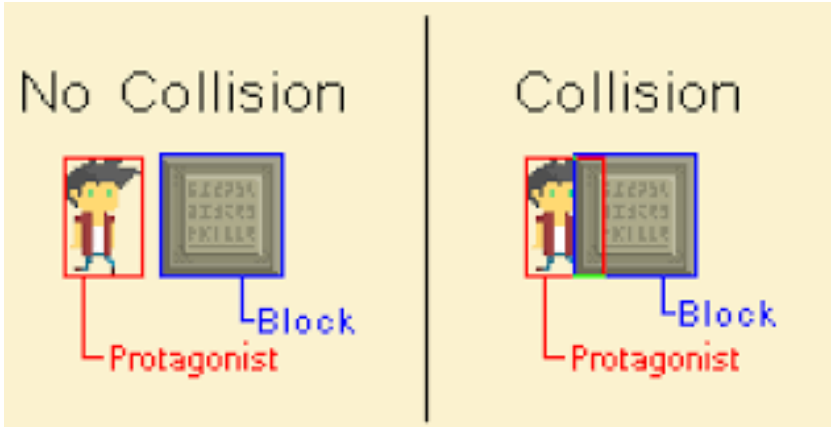
Image = Collection of Pixels (Picture Elements)

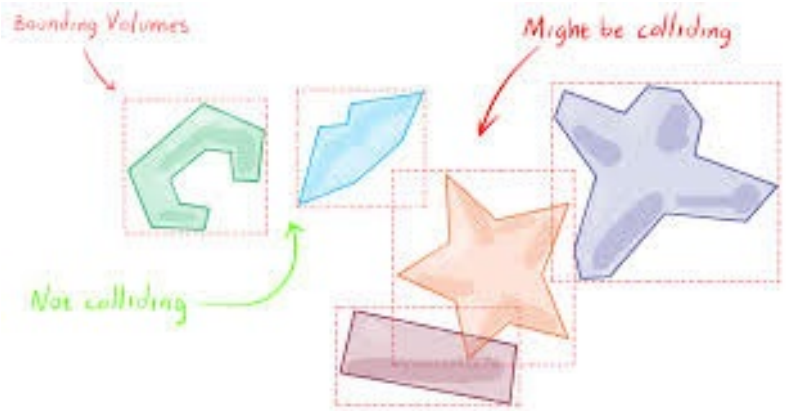


Subdividing Geometry Objects



Collision Detection (e.g., in Video Games)

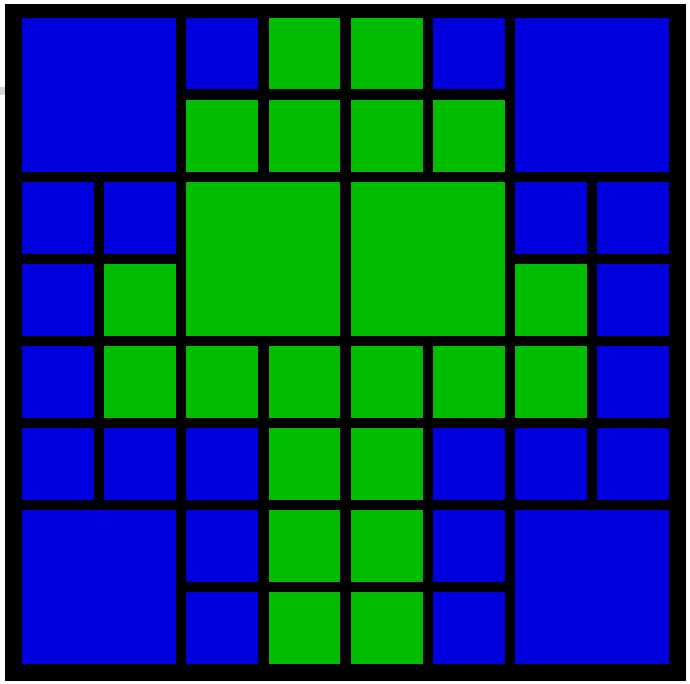




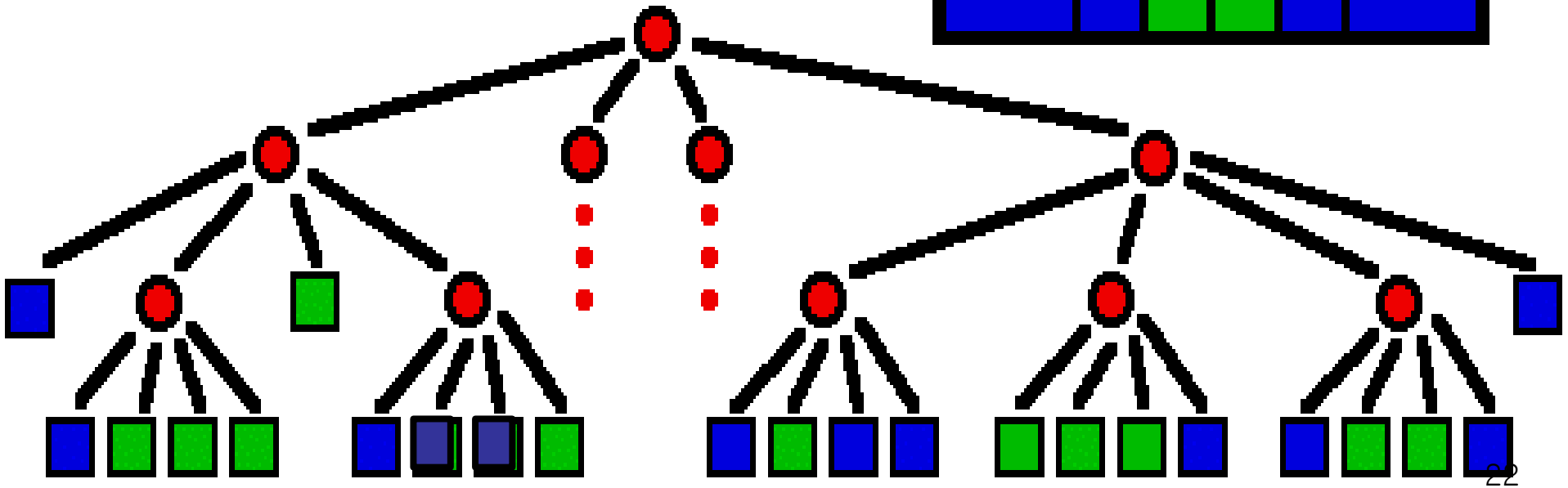
Applications: Representing an Image

- A region quadtree with a depth of n may represent an image consisting of $2^n \times 2^n$ pixels, where each pixel value is 0 or 1.
- The root node represents the entire image region. If the pixels in any region are not entirely 0s or 1s, it is subdivided.
- Each leaf node represents a block of pixels that are all 0s or all 1s.

A QuadTree for Storing 2-D Image Elements



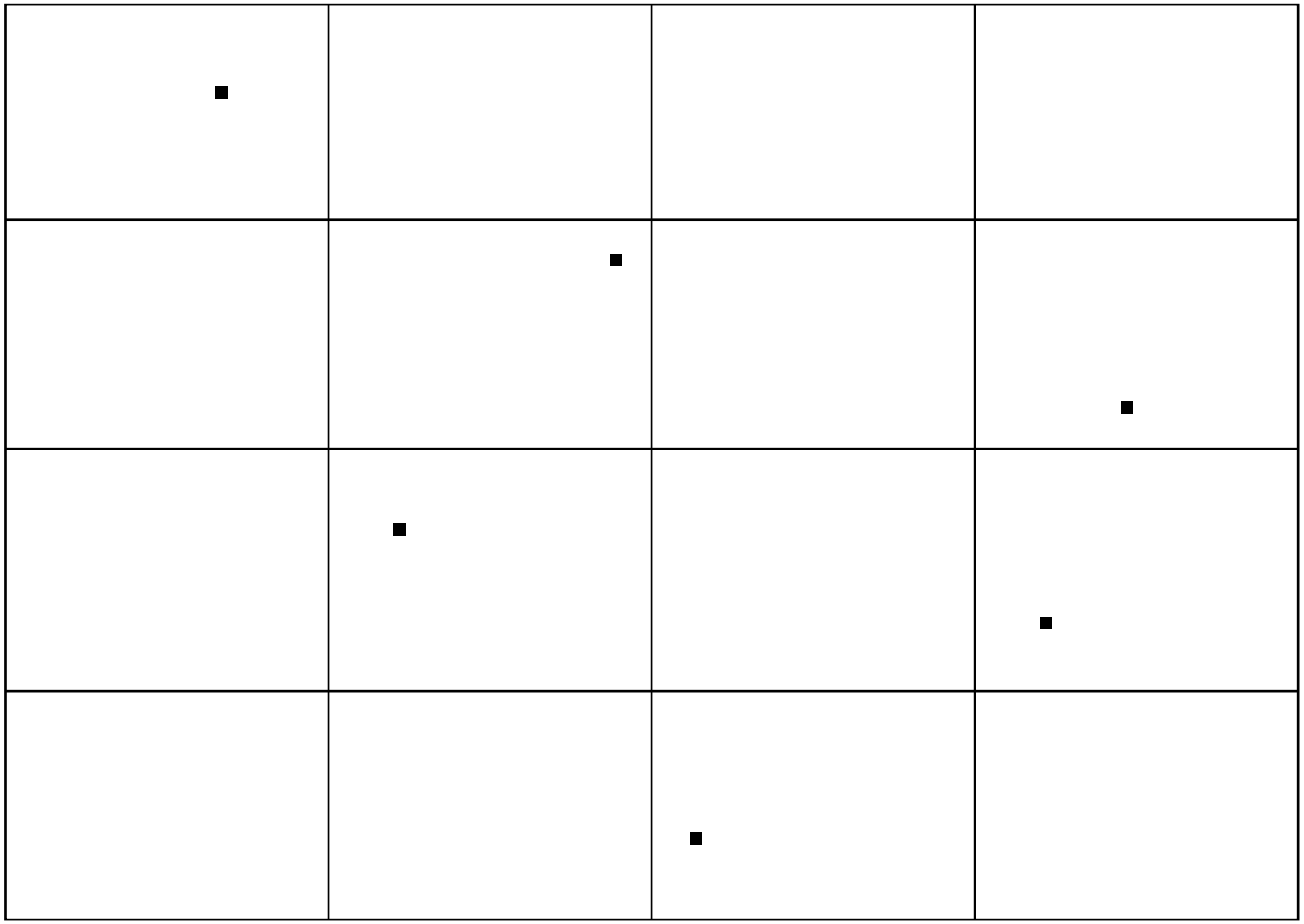
<clockwise from the 1st quadrant>



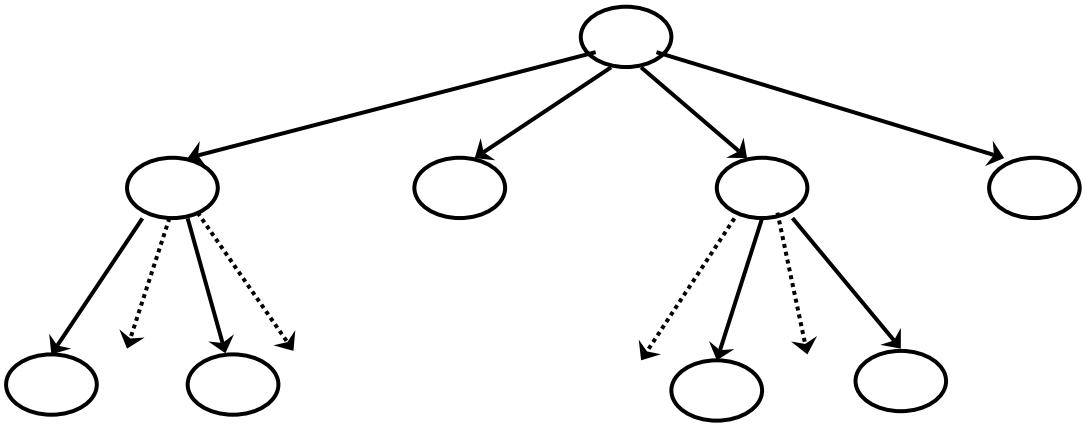
Storing and Accessing 2-D Image Elements

- Recursive subdividing of a quadrant
  - until each sub-quadrant has only one color
- Fast access to any region of an image at any level
  - any quadrant at any level of the tree
- Controlling the resolution of an image
  - raising or lowering the level of the tree

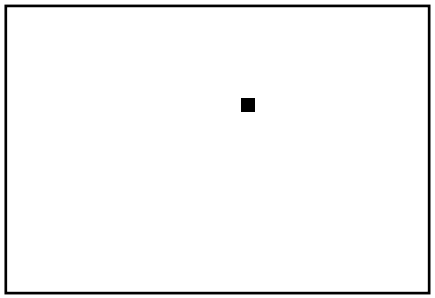
Exercise: Draw a Quad-Tree for the Following Point Data (clockwise from the 1st quadrant)



Exercise: Result



How to Determine Which Subtree to Go To?



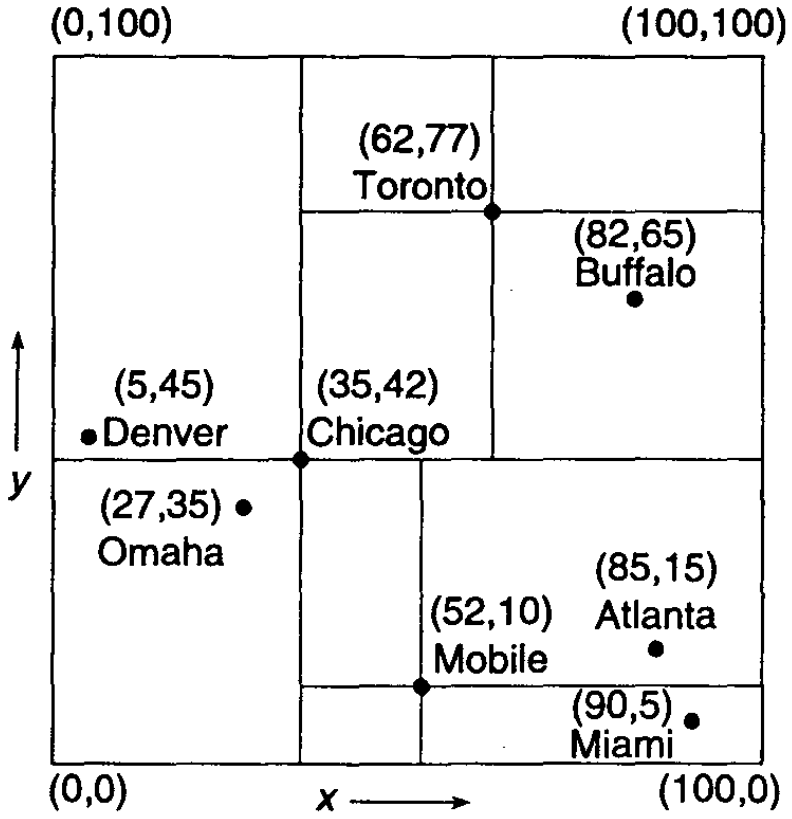
- point: (x, y)
- bounding rectangle (BR): ((x1,y1), (x2,y2))
- Each quadtree node has a corresponding BR.
- At each node, determine which of the child nodes (rectangles) can contain the search point or search rectangle.
- (The rectangle BR data may be stored in each non-leaf node or may be computed.)

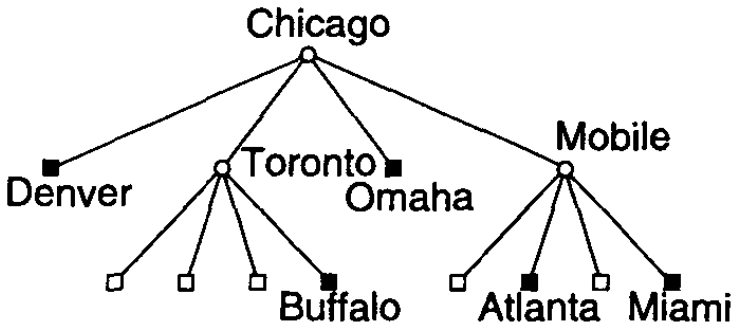
Point QuadTree

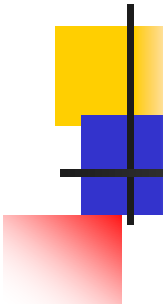
- Adaptation of a binary tree to represent 2-dimensional point data
- Each non-leaf node has 4 child nodes.
- Each node contains
  - 4 pointers (NW, NE, SW, SE)
  - key (x, y coordinates)
  - Value
- ** k-d tree is better (to be covered shortly)

Point Quadtree: Example

How do we determine the root node? (later..)





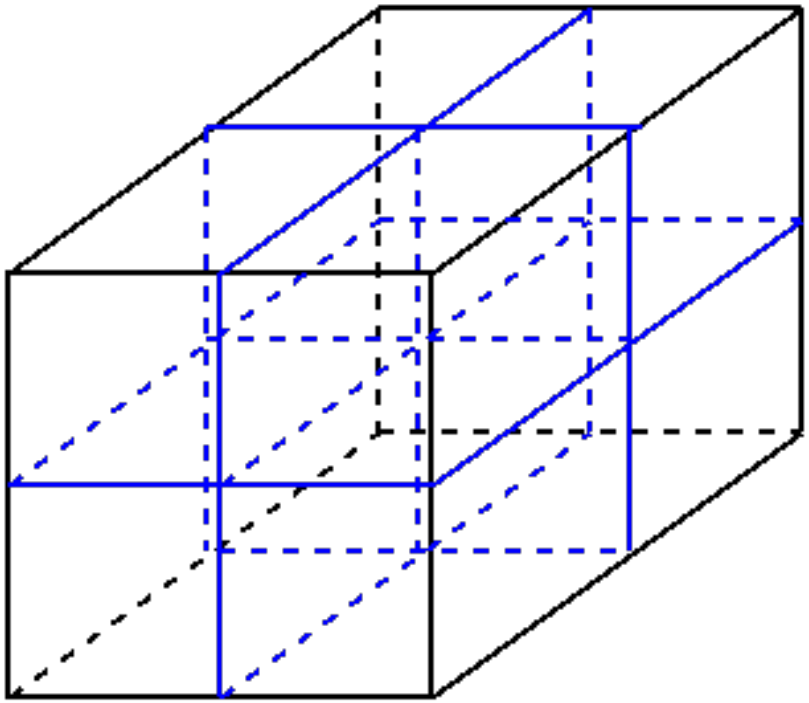


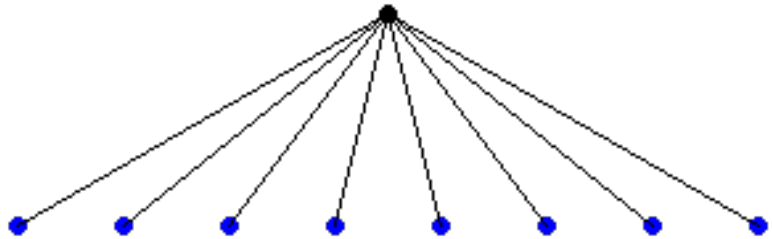
OctTree

- Tree of Degree 8
- Divides a 3-dimensional space into 8 sub-cubes
- Performance: $O(\log_8 n)$
- Leaf nodes can store pointers to 3-D geometry objects, 3-D image elements
- Reading
  - http://www.cs.berkeley.edu/~demmel/cs267/lecture26/lecture26.html

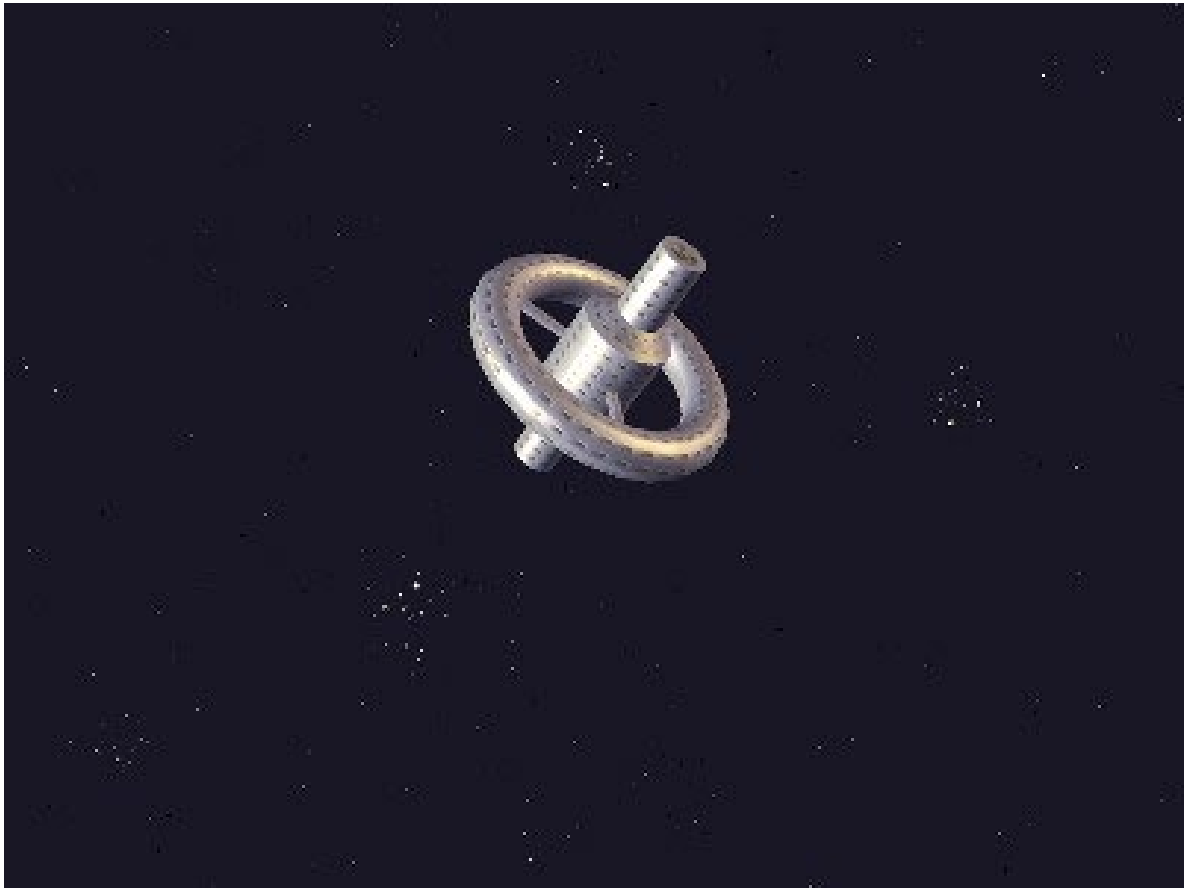
OctTree

2 Levels of an Octree



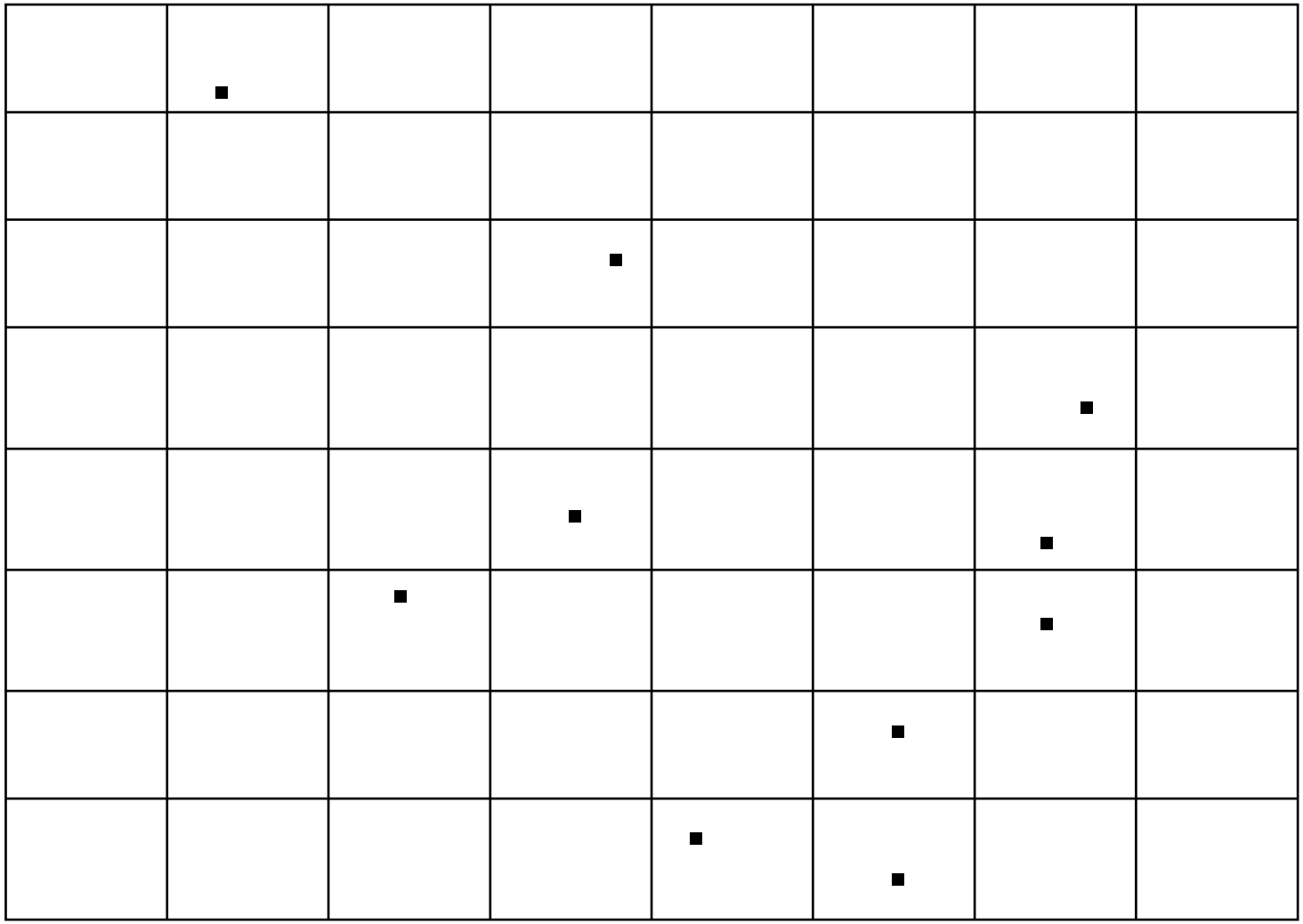


Storing and Accessing 3-D Image Elements



Exercise

Exercise: Draw a Quad-Tree for the Following Point Data (clockwise from the 1st quadrant)



End of Lecture