Q & A session for INFS 4205/7205

-- A brief review

- Fengmei Jin
- fengmei.jin@uq.edu.au

+ Spatial Data

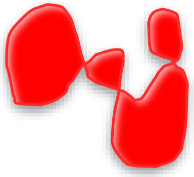
- Any data with a location component
  - 2D Space
    - Geographical space: GIS, Urban Planning
  - 3D Space
    - (x, y, z): The universe, brain model, molecule structure…
    - (x, y, t): Trajectory
- Two types of spatial data
  - Those data about the space (e.g., road networks, maps)
  - Those data about objects (e.g., location of shops, location about cars)

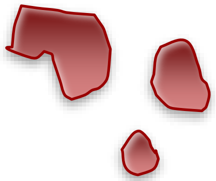
+ Spatial DBMS

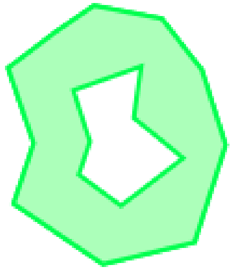
- A spatial database system is a database system
- It offers spatial data types in its data model and the query language
- It supports spatial data types in its implementation, by providing spatial indexing and efficient algorithms for spatial join queries and other types of spatial queries
  - Retrieve the spatial data without scanning the whole set

+ Spatial Data Model in Oracle

- Element, Geometry and Layer
- **Element:** the basic building block of a geometric feature
  - **Point data:** One point, stored as an (x, y) pair
  - **Line data:** Two points representing the start and the end of a line segment
  - **Polygon data:** A sequence of coordinates, one vertex pair for each line segment of the polygon
    - Simple Polygon: Both boundary and the interior
    - Complex polygons (Geometry)
      - Self-intersecting boundary
      - Multiple dis-connected components





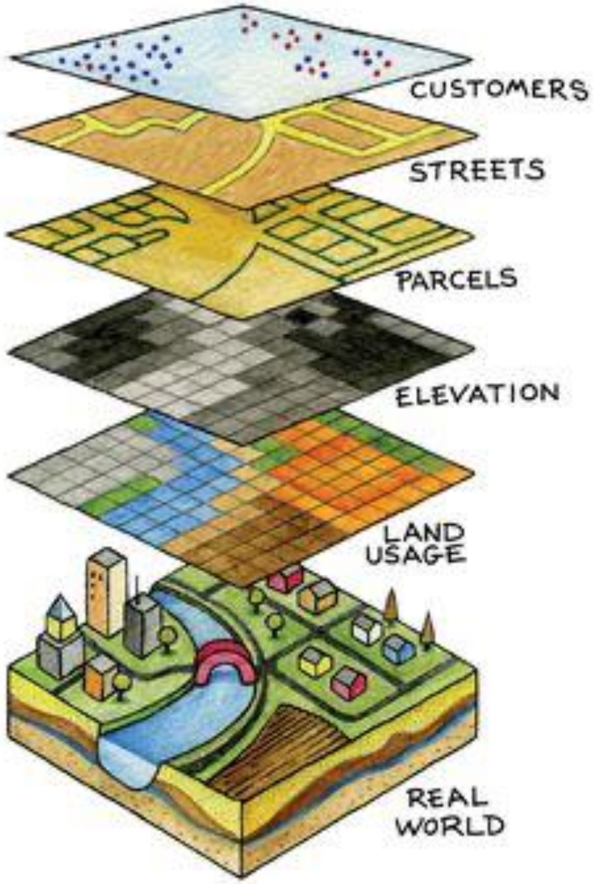


Spatial Data Model in Oracle

- **Geometry:** representation of a user’s spatial feature, modelled as an ordered **set of elements**
  - Each geometry has a unique ID, and can be associated with a set of attributes
  - A geometry might describe a lake
    - A polygon with nested polygons for islands
    - Attributes such as lake name, water capacity, fauna, flora, …

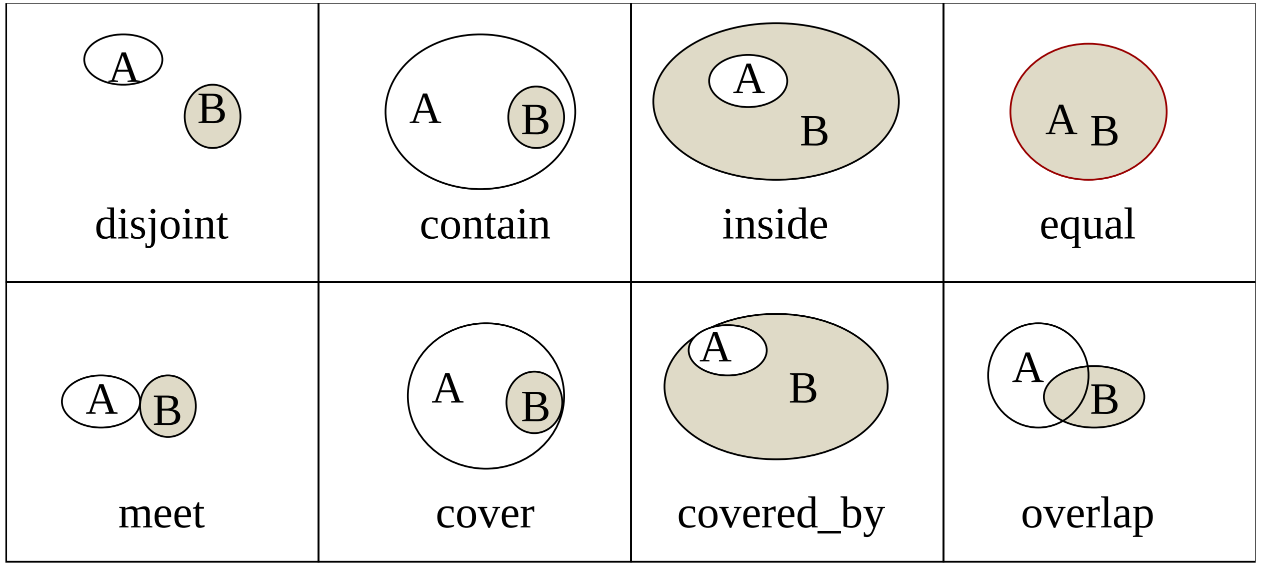
+ Spatial Data Model in Oracle

- **Layer:** a **collection of geometries** having the same attribute set
  - Examples: soil types, road network, political boundaries, population density, crops, weather conditions,…



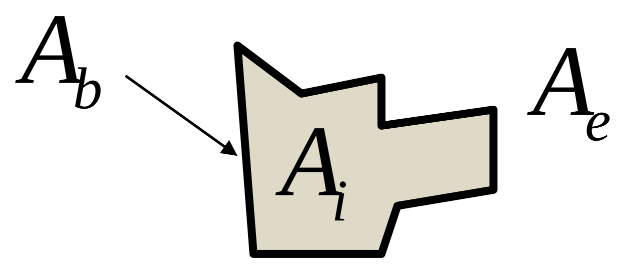
+ Spatial Relationships

- Topological
  - Invariant under translation such as rotation and scaling



- Directional
  - E.g., Above, Left, North of,…
  - May change with rotation
- Metric
  - E.g., Distance, Length, Area,…
  - May change with scaling

The 9-Intersection Matrix



A_b: boundary

A_i: interior

A_e: exterior

$$\begin{pmatrix} A_b \cap B_b & A_b \cap B_i & A_b \cap B_e \\ A_i \cap B_b & A_i \cap B_i & A_i \cap B_e \\ A_e \cap B_b & A_e \cap B_i & A_e \cap B_e \end{pmatrix}$$

Each element is either 1 or 0.

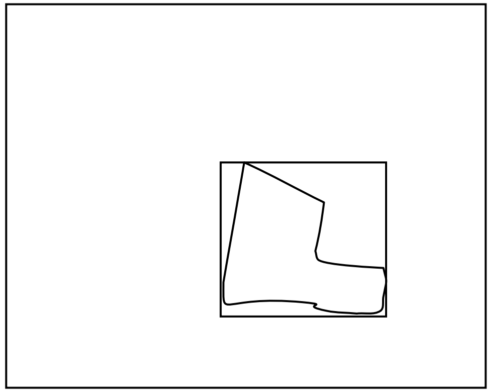
Egenhofer, M.J.; Herring, J.R. (1990). "A Mathematical Framework for the Definition of Topological Relationships"

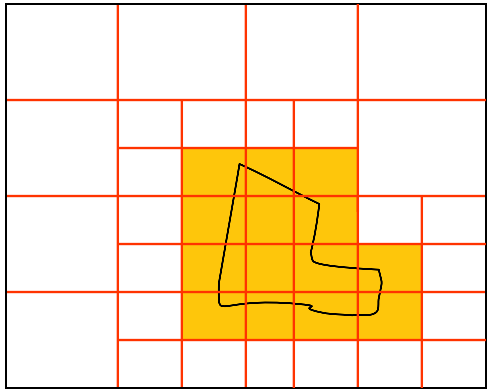
+ Spatial Indexing

- Purpose:
  - Efficiency in processing spatial selection, join and other spatial operations
- Two strategies to organize space and objects
  - Map spatial objects into 1D space and use a standard index structure (B-tree)
  - Dedicated external data structures
- Basic ideas
  - Approximation
    - Bounding box, Grids
  - Hierarchical Data Organization

+ Object Approximation

- A fundamental idea of spatial indexing is the use of approximation
- Continuous Approximation
  - Object centric
  - Example:
    - Use of MBRs (Minimum Bounding Rectangles)
    - R-Tree
- Grid Approximation
  - Space centric
    - Faster mapping
    - Uniform / Non-uniform
    - High-D?
  - Example:
    - Quad-Tree



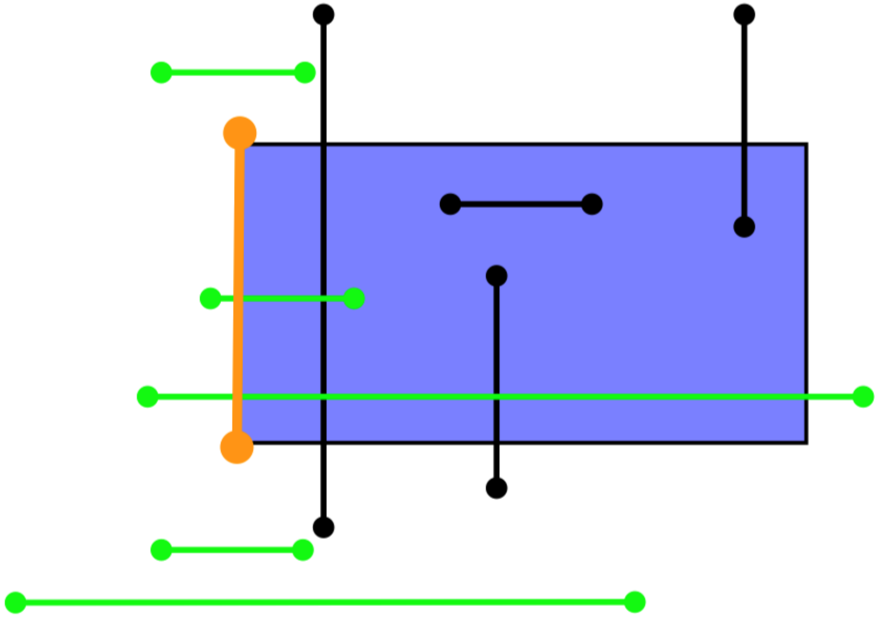


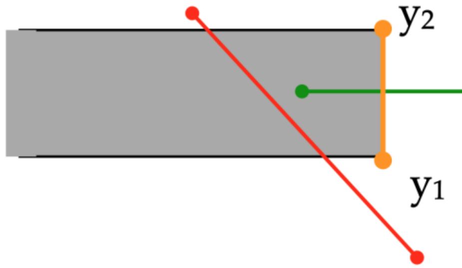
+ Line Data

- Arbitrary direction and shape
  - Use MBR and treat like polygons
  - Treated as trajectory data
- Straight line segments with ‘perpendicular directions
  - Align parallel the axis
  - Interval Tree
- Straight line segments with different directions
  - Use Point Index techniques to index the two end points separately
  - Segment Tree

+ Indexing Structures for line/segment query

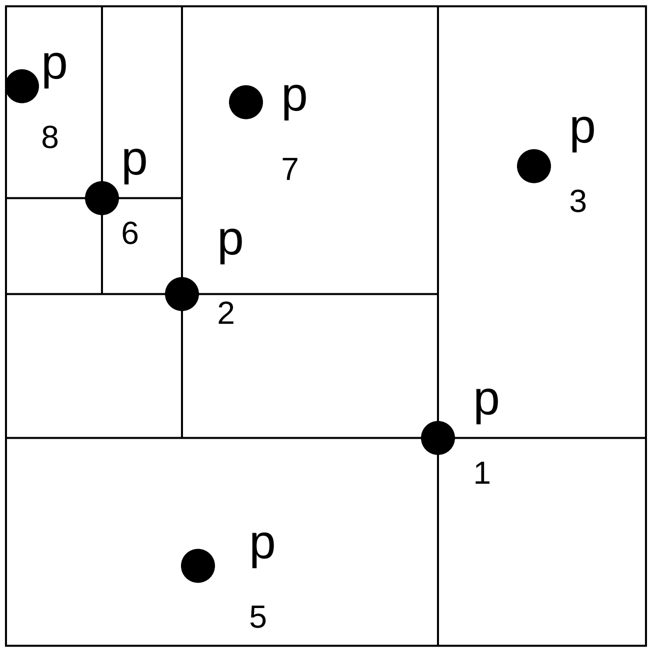
- 2D Range Tree – Binary Tree of Binary Tree
  - One Binary Tree in X (x-Tree)
  - One Binary Tree in Y for each node in the X-Tree
- Interval Tree
  - Each node stores endpoints of intervals located inside
    - Two sorted lists / 2D range tree
- Priority Search Tree
  - Good for 1-Side Range Query: $[-\infty, x], [y_1, y_2]$
    - Heap for one dimension (like x-axis)
    - Binary Search Tree for the other (y-axis)
- Segment Tree
  - Arbitrarily Oriented Segments
  - Store the segments in the Binary Search Tree

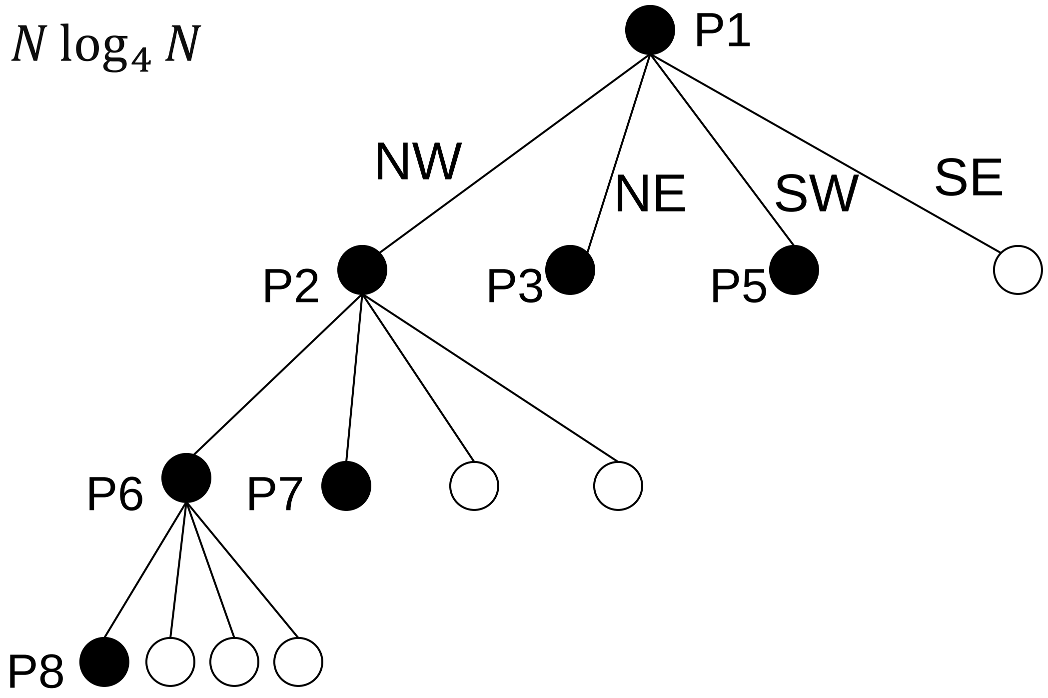




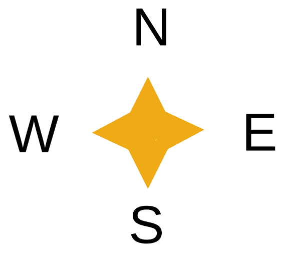
+ Point Quadtree

- Insertion
  - Random insertion roughly $N \log_4 N$



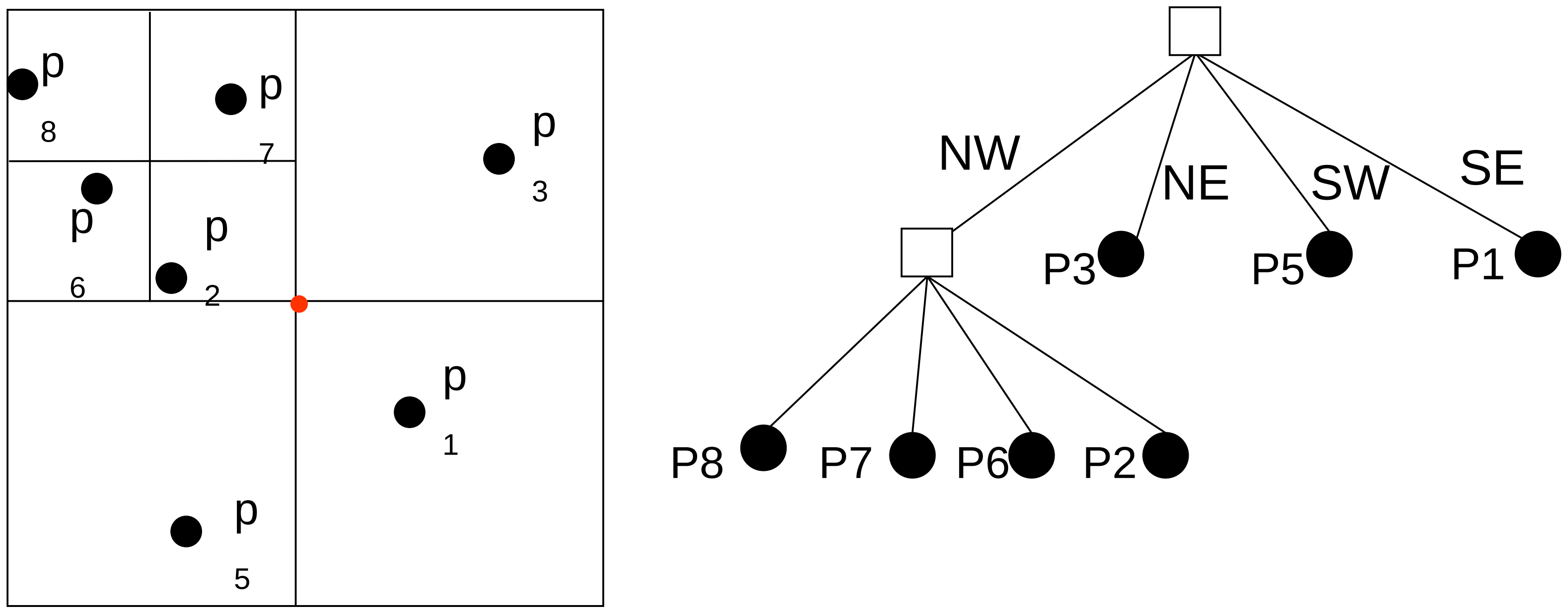


- When is the worst case?
  - Insertion takes $N(N-1)/2$



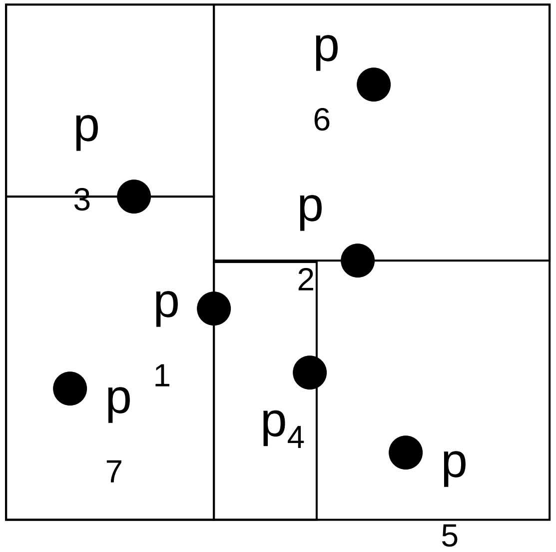
+ Region Quadtree

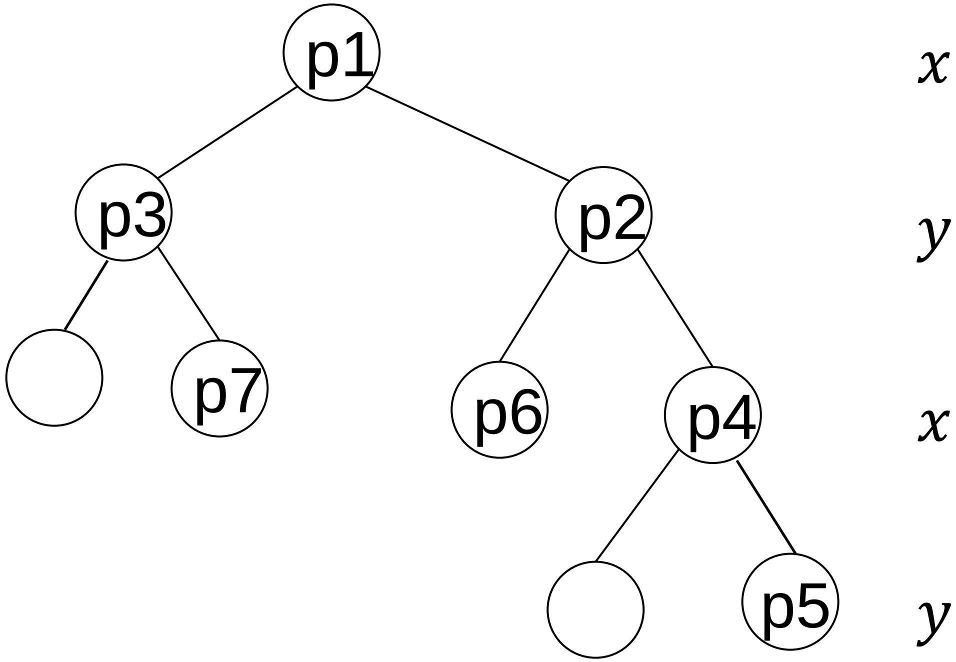
- PR Quadtree



- Based on regular decomposition of the universe
  - Recursively decomposing a region into four congruent blocks
  - Only leaves contain data

+ kd-Tree Construction



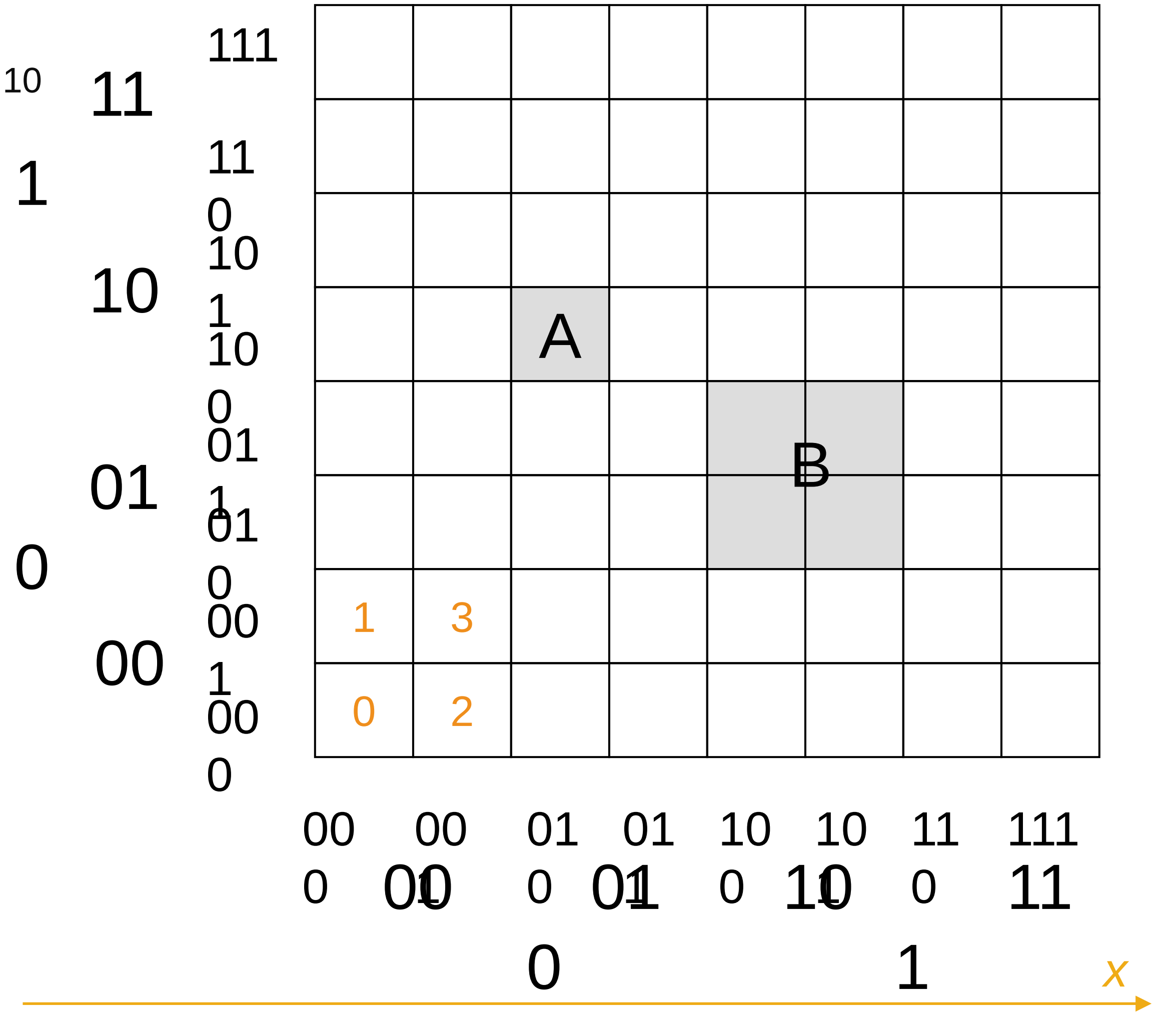


Depends on the order of insertion (not robust for sorted data).

Variations: non-alternative, data at leaves only, representing regions etc.

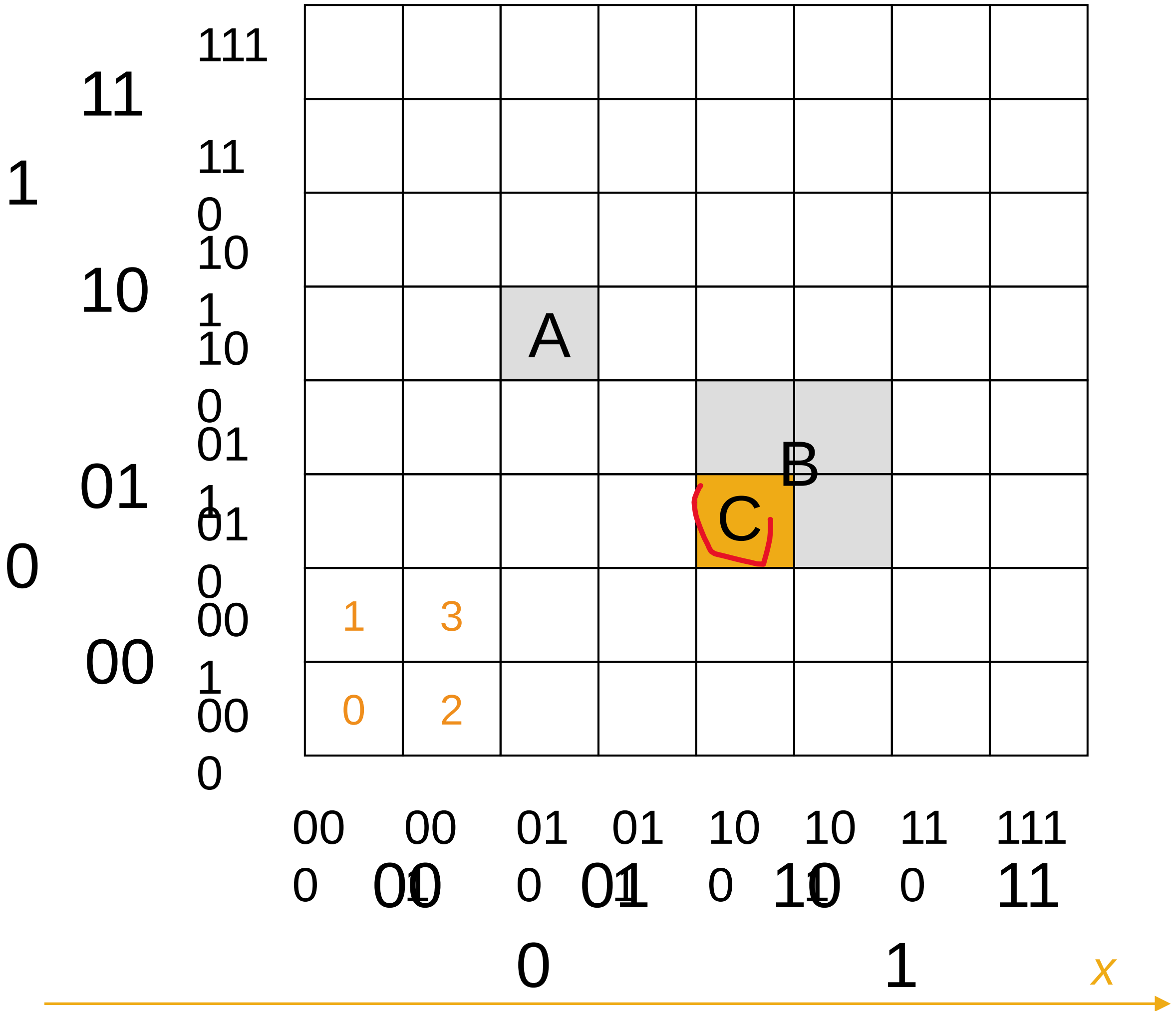
+ Z-Order

- How to obtain the z-order?
  1. Counting: A is 24
  2. Quaternary: $(120)_4 = (24)_{10}$
  3. Bit-Interleaving
     - $x_0 y_0 x_1 y_1 \ldots$
     - $(011000)_2 = (24)_{10}$
     - Works fine with varying resolutions
     - B: $(21)_4$
     - $(1001)_2$



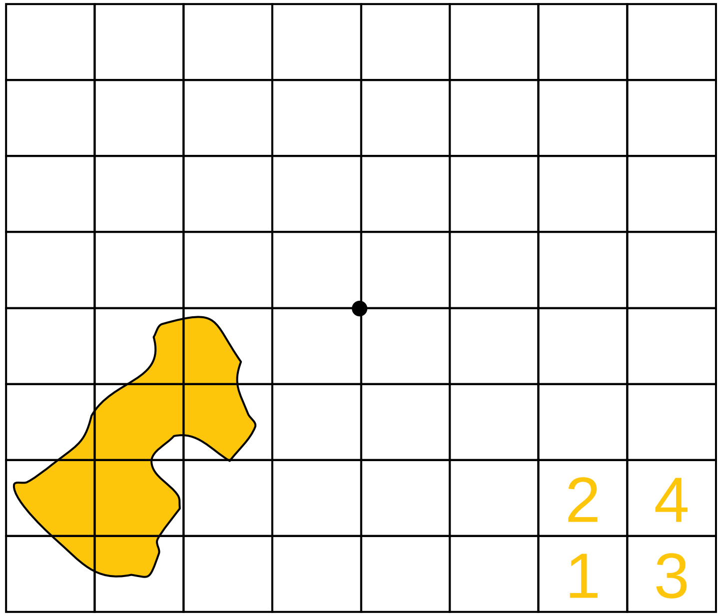
+ Z-Value Example

- B Covers C (Base 4)
  - B: 21
  - C: 210
- B covers C (Base 5)
  - B: 320
  - C: 321



+ Transformation: Using Z-Ordering

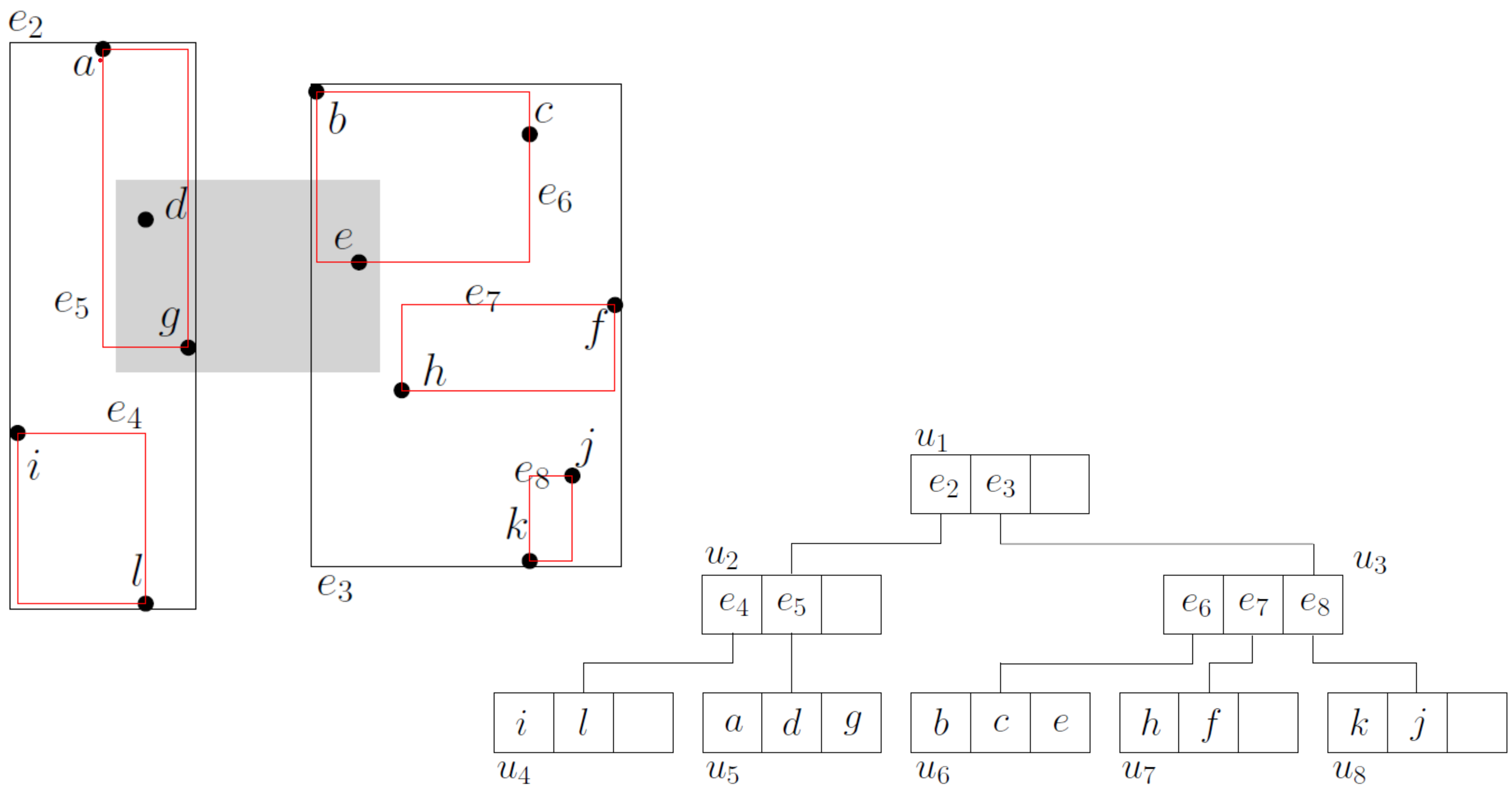
- Granularity
  - {11}, or {111, 112, 114}, or {111, 1121, 1123, 1124, 1141, 1142}
- When decomposition stops
  - Current cell either fully out or in the polygon
  - Reached the “resolution”



…the entire space is 1.

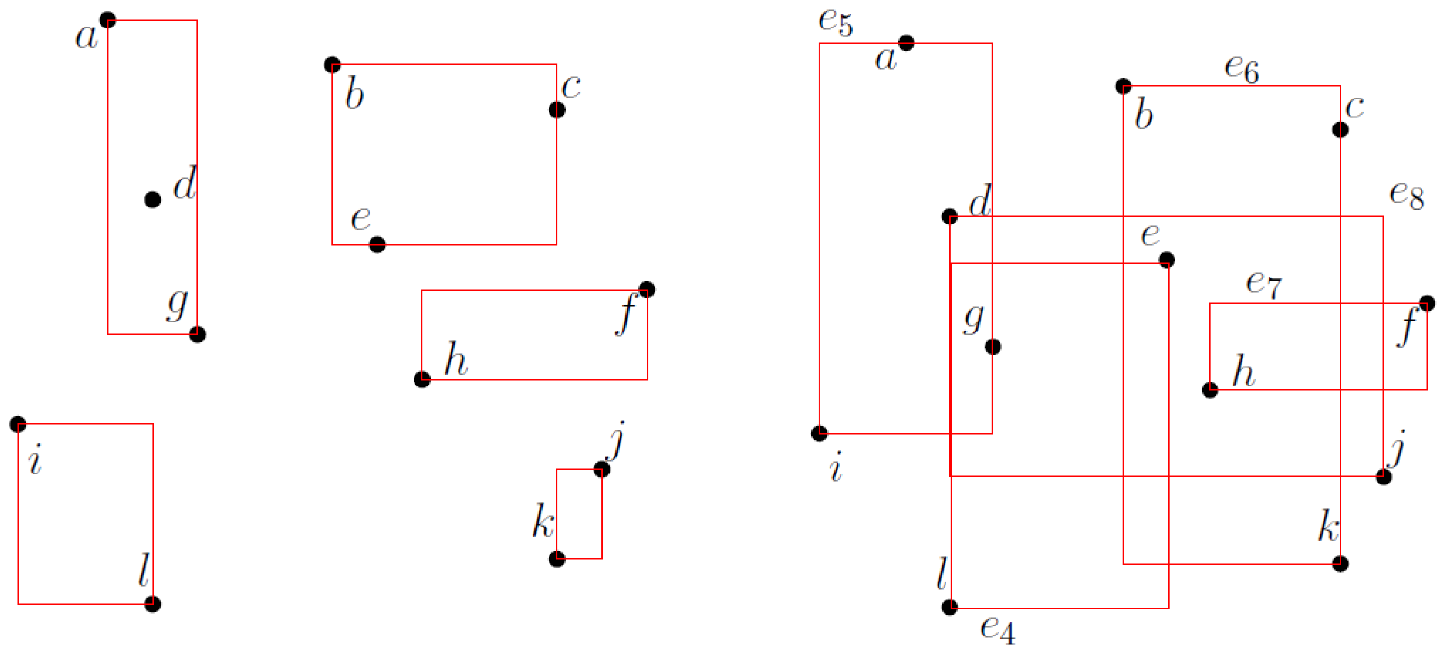
+ R-Tree Range Query

- u_1, u_2, u_3, u_5, u_6 are accessed



+ R-Tree Construction

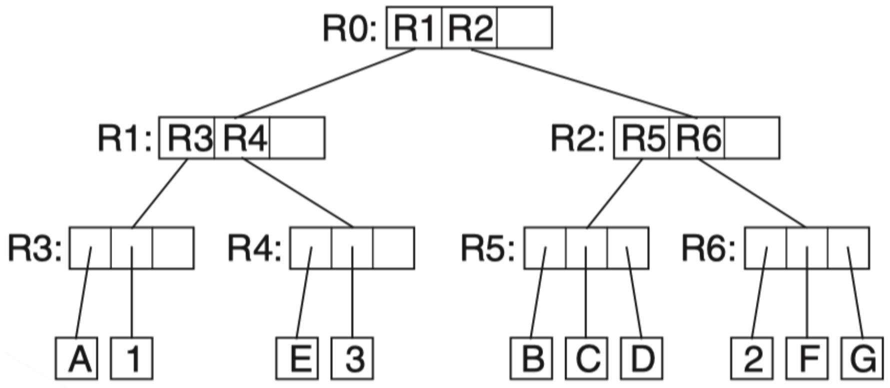
- R-Tree construction can be "arbitrary"
  - Bottom-up
  - No formal constraint on the grouping of data into nodes

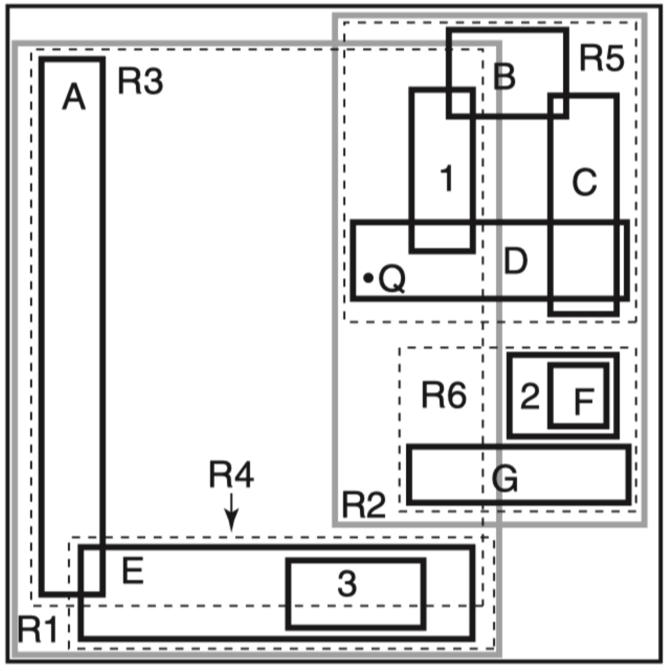


  - The left tree has a smaller perimeter sum than the right one

+ Clipping

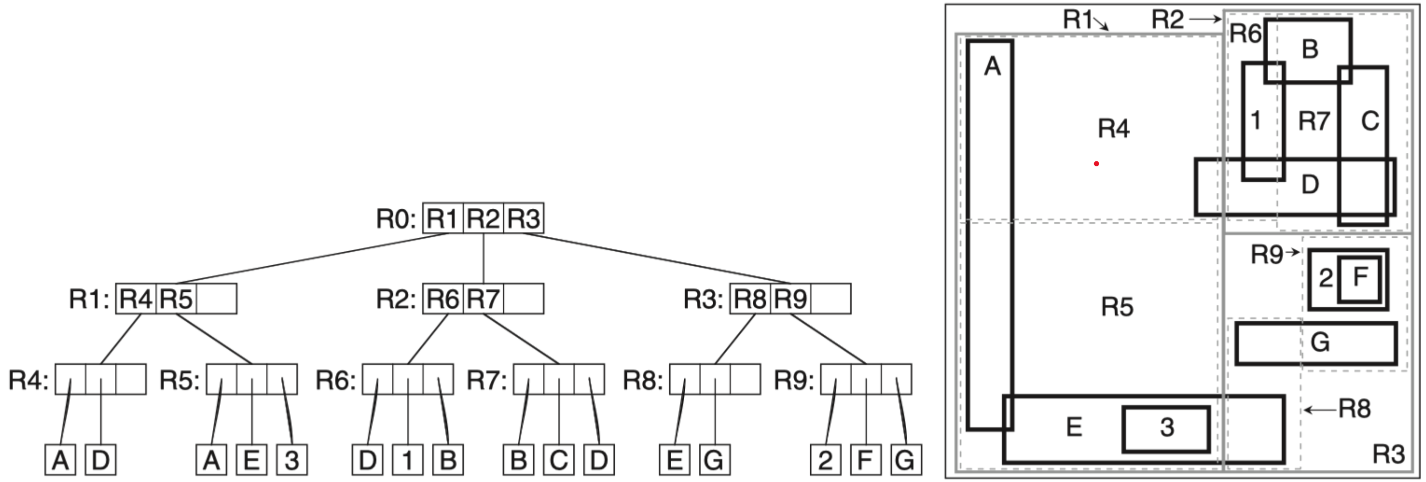
- Motivation
  - R-Tree: May examine all the MBRs at all levels
    - Because the MBR may overlap, the space is not disjointly decomposed
      - Query point Q in the example
  - Single search path for a point query





+ R$^+$-Tree

- Basic ideas
  - A hierarchy of overlapping MBRs → A hierarchy of disjoint MBRs
    - Regular grid / Irregular grid
  - Clipping polygon at cell boundaries
    - Whenever an MBR at a lower level overlaps with another MBR, decompose it into a collection of non-overlapping sub-MBRs
  - Allowing one polygon in multiple cells
    - Non-overlapping is achieved at the cost of space

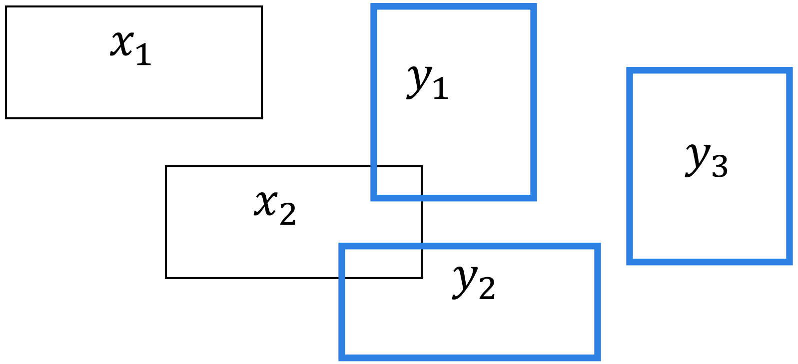


+ Filter-and-Refine

- A most commonly used processing strategy
- Motivation
  - Avoid expensive spatial processing as much as possible
- Basic Idea
  - A filter step, followed by a refinement step
  - Filter step: applying *simple* operations on *approximations* of spatial objects
  - Refinement step: applying the *actual* spatial operations on the *full geometry* of spatial objects

+ Spatial Join Example

- Intersection join



- Join results: (x_2, y_1), (x_2, y_2)
- Other spatial join operations
  - Topological: intersection, adjacent, contains…
  - Metrical or directional: within_distance…
  - More advanced: nearest....

+ Processing Framework

- Filter step
  - Find a set of candidates $C = \{(p, s): p \in R\ and\ s \in S\}$ *quickly*
    - Using approximations (e.g., MBR) and indexes
    - Other filter steps possible (eg, using *progressive* approximation)
- Housekeeping step
  - Process C such that the IO cost for the refinement step can be further minimized
  - E.g., Removing duplicates; Performing refinement in optimal order
- Refine step
  - Fetch full geometry for the objects in each candidate, and apply a full test to drop “false hits”

+ Simple Nested-Loops Join


```
for each r in R
    for each s in S
        if (r,MBR intersect s.MBR)
            put (r, s) to the candidate set;
```

+ Indexed Nested-Loops Join

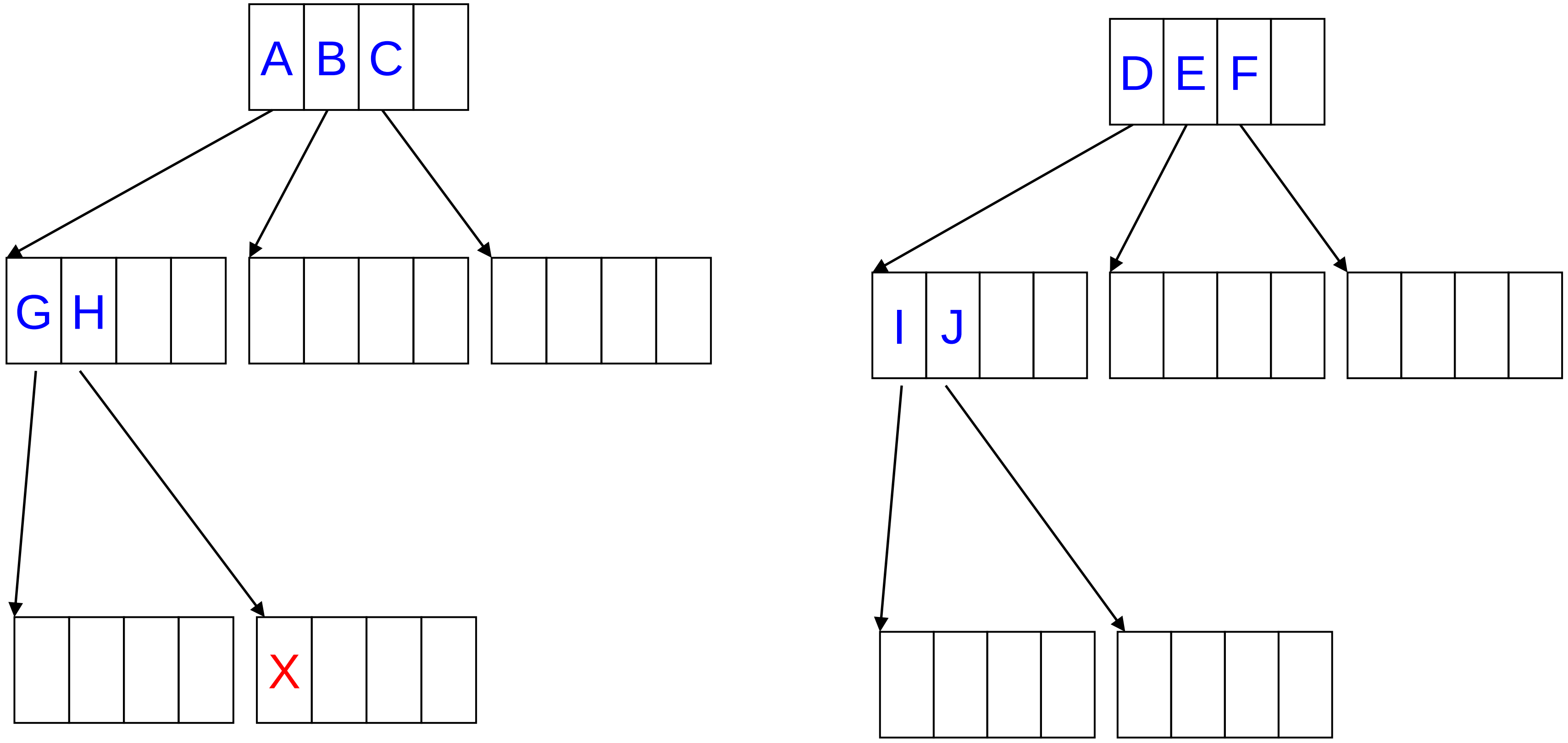
- Using a window query against S

for each *r* in *R*

Find all *s* in *S* such that *s.MBR* **intersect** *r.MBR*

put (*r*, *s*) to the candidate set;

+ Nested-Loops With R/R+-Trees



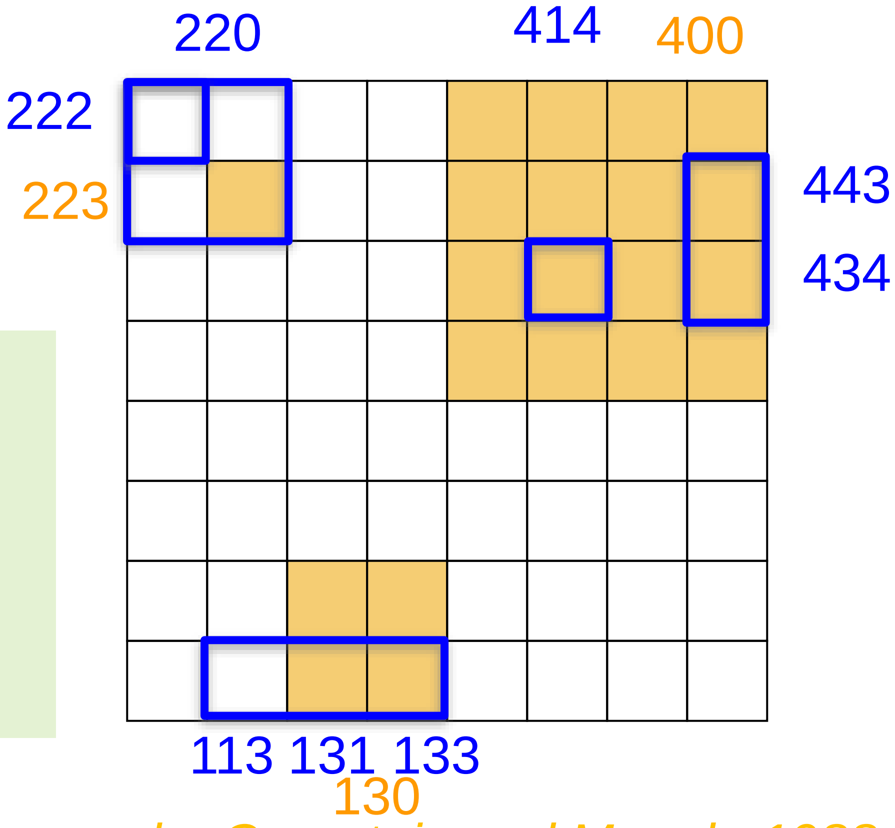
Is it possible to produce the same candidate (p_1, p_2) multiple times?

Reading: the paper by Brinkhoff, Kriegel and Seeger 1993.

+ Sort-Merge Join using Z-values

r	*s*
~~113~~	130
131	223
133	400
220	
222	
414	
434	
443	

- Differences
  1. $131 \neq 130$
  2. Cannot move from 130 to 223 immediately



Algorithm Sketch:

1) Two sorted lists and two pointers
2) Synchronized traversal
   - overlap(*r*, *s*)?
   - increase min(*r*, *s*)
3) Some values in <u>stack</u>

Reading: the paper by Orenstein and Manola 1988.