

Spatial Storage and Indexing

UNIVERSITY OF MINNESOTA
Driven to DiscoverSM



Outline

- Recap:
 - Physical data model
 - Physical data organization (file structures)
- Space filling curves, e.g., Z-curve and Hilbert curve
- Spatial index structures, e.g., grid, R-tree, and quad-tree

RECAP

Physical Model - Analogy with Vehicles

- Logical models:
 - Car: accelerator pedal, steering wheel, brake pedal, ...
 - Bicycle: pedal forward to move, turn handle, pull brakes on handle
- Physical models :
 - Car: engine, transmission, master cylinder, break lines, brake pads, ...
 - Bicycle: chain from pedal to wheels, gears, wire from handle to brake pads
- We now go, so to speak, “under the hood”



Physical Model - Motivational Query

- Input

Given: (1) my location L ,

(2) 100 millions spatial facilities (restaurants, schools, hotels, etc)

Output

Find the nearest 3 facilities to my location L



Physical Model - Motivational Query

- Input

Given: (1) my location L,

(2) 100 millions spatial facilities (restaurants, schools, hotels, etc)

Output

Find the nearest 3 facilities to my location L

- Logical models:

- Express query in a high-level language, what are the operators?
- Ex.: `SELECT * FROM Facilities F WHERE Nearest(3, F.Loc, L)`



Physical Model - Motivational Query

- Input

Given: (1) my location L,

(2) 100 millions spatial facilities (restaurants, schools, hotels, etc)

Output

Find the nearest 3 facilities to my location L

- Logical models:

- Express query in a high-level language, what are the operators?
- Ex.: `SELECT * FROM Facilities F WHERE Nearest(3, F.Loc, L)`

- Physical models:

- How spatial objects are physically stored and organized in files?
- How spatial objects are retrieved from files?



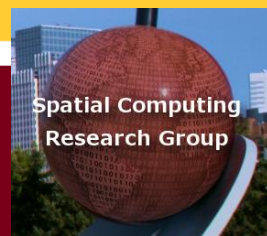
Physical Model - Motivational Query

- Input: Given my location L , and 100 millions spatial facilities
Output: Find the nearest 3 facilities to my location L
- Physical models examples:
 - One way:
store all objects in one file in random order, then scan all objects to get the nearest 3 objects to L



Physical Model - Motivational Query

- Input: Given my location L , and 100 millions spatial facilities
Output: Find the nearest 3 facilities to my location L
- Physical models examples:
 - One way:
store all objects in one file in random order, then scan all objects to get the nearest 3 objects to L
 - Another way:
store multiple files, each file contains portion of data, then locate files relevant to L to retrieve objects



What is a Physical Data Model of a Database?

- Concepts to implement logical data model
- Using current components, e.g. computer hardware, operating systems
- In an efficient and fault-tolerant manner



Why Learn Physical Data Model Concepts?

- To be able to choose between DBMS brand names
 - Some brand names do not have spatial indices!
- To be able to use DBMS facilities for performance tuning
- For example, If a query is running slow,
 - One may create an index to speed it up
- For example, if loading of a large number of tuples takes for ever
 - One may drop indices on the table before the inserts
 - And recreate index after inserts are done!



Concepts in a Physical Data Model

- Database concepts
 - Conceptual data model - entity, (multi-valued) attributes, relationship, ...
 - Logical model - relations, atomic attributes, primary and foreign keys
 - Physical model - secondary storage hardware, file structures, indices, ...



Concepts in a Physical Data Model - Examples from Relational DBMS

- Secondary storage hardware: Disk drives
- File structures - sorted
- Auxiliary search structure -
 - Search trees (hierarchical collections of one-dimensional ranges)



An Interesting Fact about Physical Data Model

- Physical data model design is a trade-off between
 - Efficiently support a small set of basic operations of a few data types
 - Simplicity of overall system
- Each DBMS physical model
 - Choose a few physical DM techniques
 - Choice depends chosen sets of operations and data types



An Interesting Fact about Relational DBMS Physical Model

- Data types: numbers, strings, date, currency
 - One-dimensional, totally ordered
- Operations:
 - Search on one-dimensional totally order data types
 - Insert, delete, ...



Physical Data Model for SDBMS

- Is relational DBMS physical data model suitable for spatial data?
 - Relational DBMS has simple values like numbers
 - Sorting, search trees are efficient for numbers
 - These concepts are not natural for Spatial data (e.g. points in a plane)
- Reusing relational physical data model concepts
 - Space filling curves define a total order for points
 - This total order helps in using ordered files, search trees
 - But may lead to computational inefficiency!



Physical Data Model for SDBMS - New Spatial Techniques

- Spatial indices, e.g. grids, hierarchical collection of rectangles
- Provide better computational performance



Common assumptions for SDBMS physical model - Spatial Data

- Dimensionality of space is low, e.g. 2 or 3
 - Data types: OGIS data types
- Approximations for extended objects (e.g. linestrings, polygons)
 - Minimum Orthogonal Bounding Rectangle (MOBR or MBR)
 - $MBR(O)$ is the smallest axis-parallel rectangle enclosing an object O
- Supports filter and refine processing of queries



Common Spatial Queries

- Physical model provides simpler operations needed by spatial queries!
- Common Queries
 - Range query: Find all objects within a query rectangle.
 - k-nearest neighbor: Find k points closest to a query point.
 - Join query: Find all intersecting objects from two spatial datasets.



Common Spatial Operations

- **Common operations across spatial queries**
 - Find : retrieve records satisfying a condition on attribute(s)
 - Findnext : retrieve next record in a dataset with total order
 - After the last one retrieved via previous find or findnext
 - Nearest neighbor of a given object in a spatial dataset



Storage Hierarchy in Computers - Types of Storage Devices

- Main memories - fast but content is lost when power is off
- Secondary storage - slower, retains content without power
- Tertiary storage - very slow, retains content, very large capacity



Storage Hierarchy in Computers - DBMS Usually Manage Data

- On secondary storage, e.g. disks
- Use main memory to improve performance
- User tertiary storage (e.g. tapes) for backup, archival etc.



Software View of Disks: Fields, Records and File

- Views of secondary storage (e.g. disks)
 - Data on disks is organized into fields, records, files



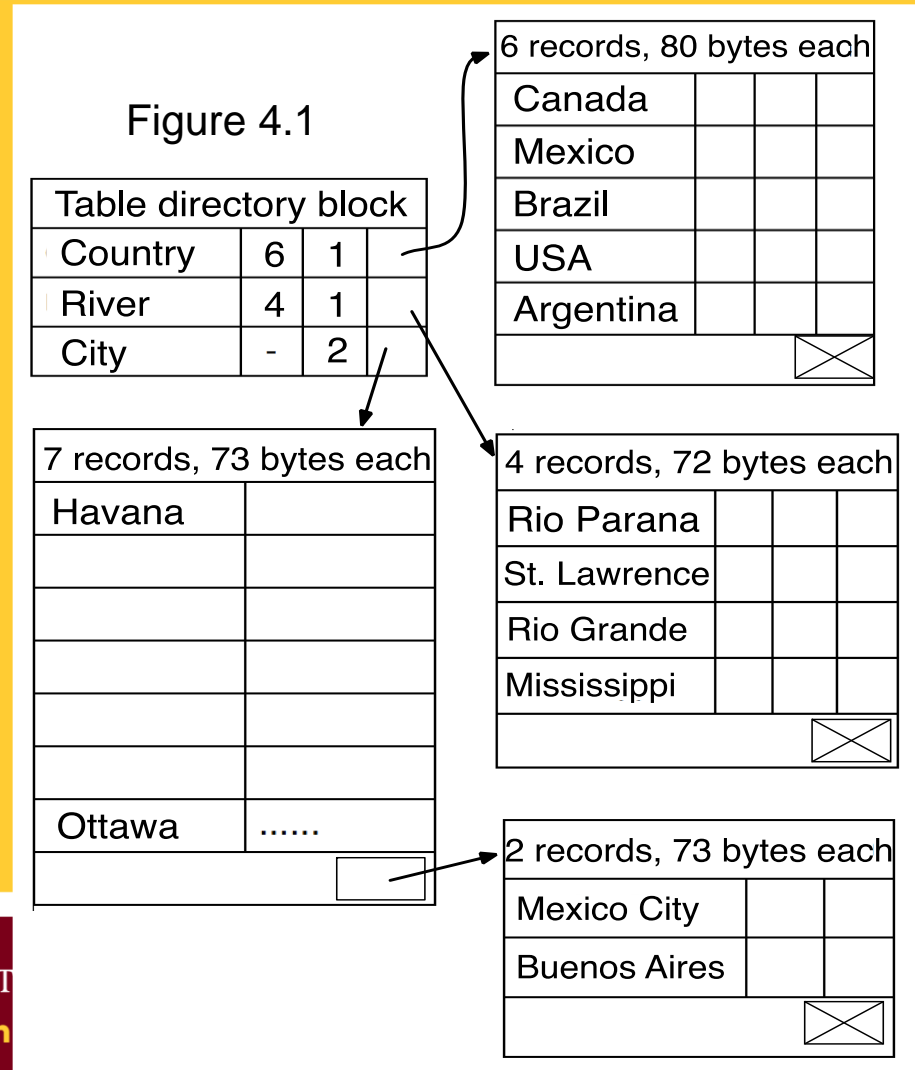
Software View of Disks: Fields, Records and File - Concepts

- Field presents a property or attribute of a relation or an entity
- Records represent a row in a relational table
 - Collection of fields for attributes in relational schema of the table
- Files are collections of records
 - Homogeneous collection of records may represent a relation
 - Heterogeneous collections may be a union of related relations



Mapping Records and Files to Disk

- Records
 - Often smaller than a sector
 - Many records in a sector
- Files with many records
 - Many sectors per file
- File system
 - Collection of files
 - Organized into directories
- Mapping tables to disk
 - Figure 4.1
 - City table takes 2 sectors
 - Others take 1 sector each



File Structures

- What is a file structure?
 - A method of organizing records in a file
 - For efficient implementation of common file operations on disks
 - Example: ordered files
- Measure of efficiency
 - I/O cost: Number of disk sectors retrieved from secondary storage
 - CPU cost: Number of CPU instruction used
 - See Table 4.1 for relative importance of cost components
 - Total cost = sum of I/O cost and CPU cost



Common File Structures

- Heap or unordered or unstructured
- Ordered
- Hashed



File Structures - Common File Operations

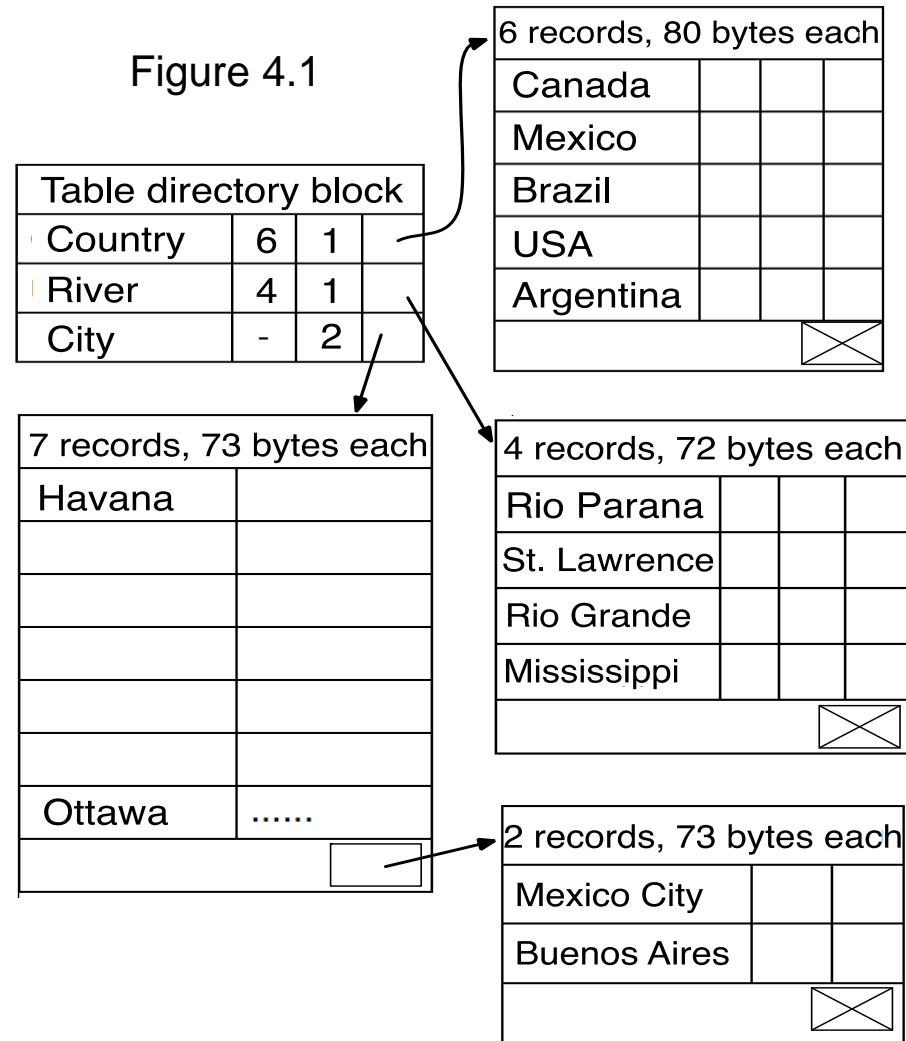
- Find: key value --> record matching key values
- Findnext --> Return next record after find if records were sorted
- Insert --> Add a new record to file without changing file-structure
- Nearest neighbor of a object in a spatial dataset



File Structures - Examples Using Figure 4.1

- Find(Name = Canada) on Country table returns record about Canada
- Findnext() on Country table returns record about Cuba
 - Since Cuba is next value after Canada in sorted order of Name
- Insert(record about Panama) into Country table
 - Adds a new record
 - Location of record in Country file depends on file-structure
- Nearest neighbor Argentina in country table is Brazil

Figure 4.1



Common File Structures

- Heap or unordered or unstructured
- Ordered
- Hashed



Basic Comparison of Common File Structures

- Heap file is efficient for inserts and used for log files
 - But find, findnext, etc. are very slow
- Hashed files are efficient for find, insert, delete etc.
 - But findext is very slow
- Ordered file organization are very fast for findnext
 - And pretty competent for find, insert, etc.



File Structures: Heap

- Records are in no particular order (Example: Figure 4.1)
- Insert can simple add record to the last sector
- Find, findnext, nearest neighbor scan the entire files

Figure 4.1

Table directory block			
Country	6	1	
River	4	1	
City	-	2	

6 records, 80 bytes each			
Canada			
Mexico			
Brazil			
USA			
Argentina			
			⊗

7 records, 73 bytes each	
Havana	
Ottawa

4 records, 72 bytes each			
Rio Parana			
St. Lawrence			
Rio Grande			
Mississippi			
			⊗

2 records, 73 bytes each			
Mexico City			
Buenos Aires			
			⊗

File Structures: Ordered

- Records are sorted by a selected field (Example Fig. 4.3 below)
- Findnext can simply pick up physically next record
- Find, insert, delete may use binary search, is very efficient
- Nearest neighbor processed as a range query

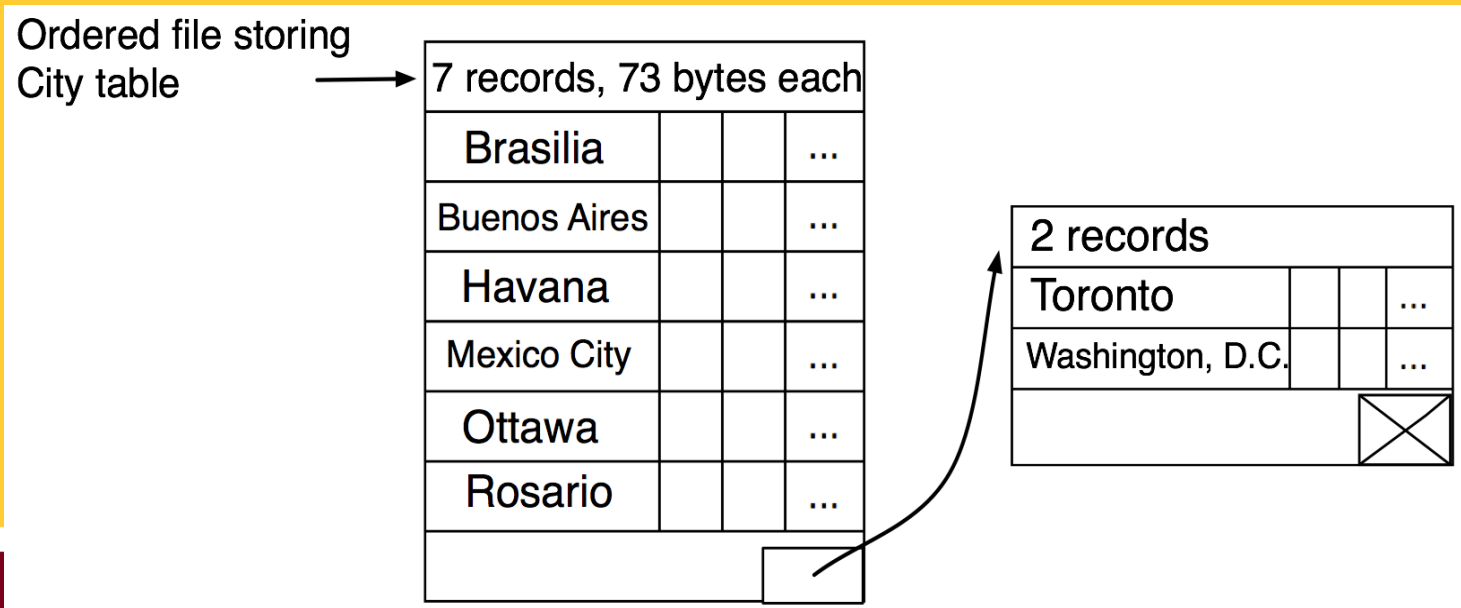
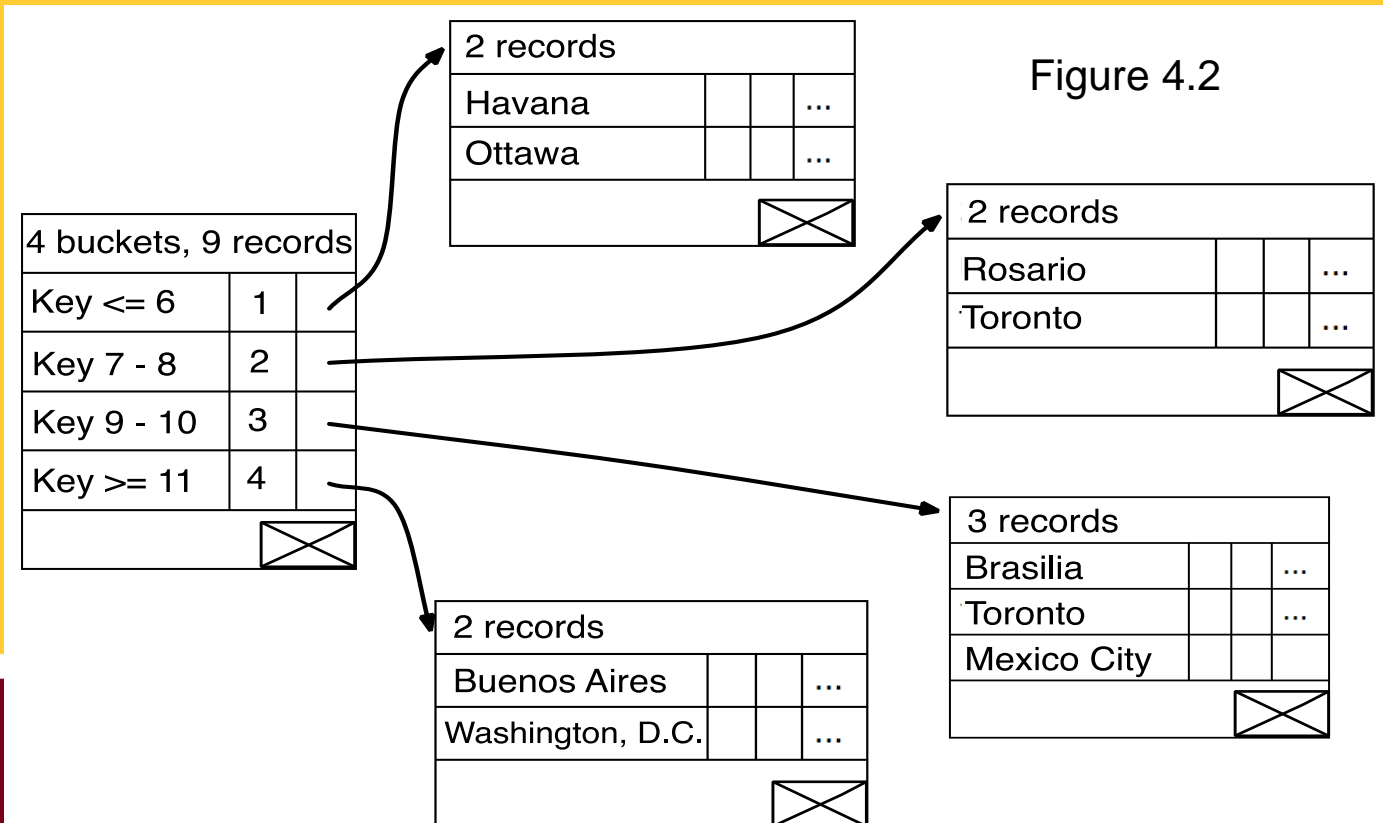


Figure 4.3

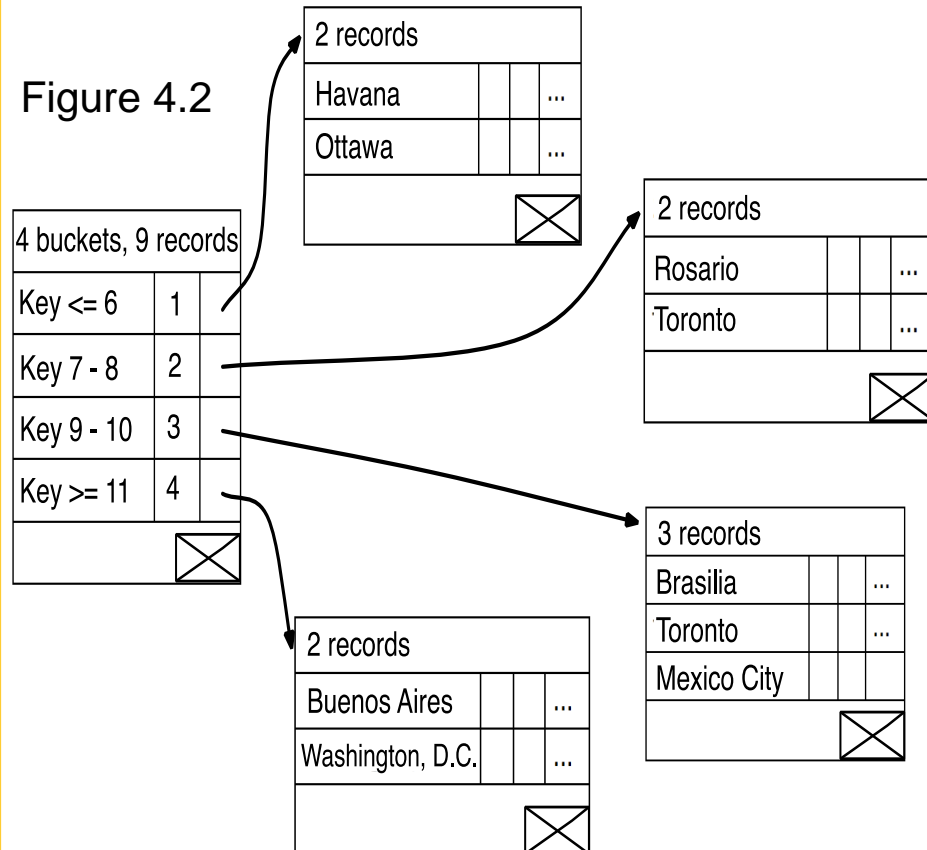
File Structures: Hash - Components

- A set of buckets (sectors)
- Hash function : key value --> bucket
- Hash directory: bucket --> sector



File Structures: Hash - Operations

- Find, insert, delete are fast
 - Compute hash function
 - Lookup directory
 - Fetch relevant sector
- Findnext, nearest neighbor are slow
 - No order among records



RECAP HIGHLIGHTS

Concepts in a Physical Data Model

- Database concepts
 - Conceptual data model - entity, (multi-valued) attributes, relationship, ...
 - Logical model - relations, atomic attributes, primary and foreign keys
 - Physical model - secondary storage hardware, file structures, indices, ...



File Structures

- What is a file structure?
 - A method of organizing records in a file
 - For efficient implementation of common file operations on disks
 - Example: ordered files
- Measure of efficiency
 - I/O cost: Number of disk sectors retrieved from secondary storage
 - CPU cost: Number of CPU instruction used
 - See Table 4.1 for relative importance of cost components
 - Total cost = sum of I/O cost and CPU cost



Common File Structures

- Heap or unordered or unstructured
- Ordered
- Hashed



Space Filling Curves

- How to order country names?



Space Filling Curves

- How to order country names?
 - Alphabetical order (total order, linear order)



Space Filling Curves

- How to order country names?
 - Alphabetical order (total order, linear order)
- How to order country spatial locations?



Space Filling Curves

- How to order country names?
 - Alphabetical order (total order, linear order)
- How to order country spatial locations?
 - There is no linear order
 - There is no total order



Space Filling Curves

- How to order country names?
 - Alphabetical order (total order, linear order)
- How to order country spatial locations?
 - There is no linear order
 - There is no total order
- Why do we need to order?



Space Filling Curves

- How to order country names?
 - Alphabetical order (total order, linear order)
- How to order country spatial locations?
 - There is no linear order
 - There is no total order
- Why do we need to order?
 - Speed-up searching
 - Use binary search, which is faster than linear search



Space Filling Curves

- Space Filling Curves (Geo-hashing)
 - Discretized two-dimensional space
 - Suggest a sorting order on points
 - Ex.: row-major, column-major, Z-curve, Hilbert-curve, ...



Space Filling Curves

- Naïve:

	0	1	2	3
0	0	1	2	3
1	7	6	5	4
2	8	9	10	11
3	15	14	13	12



Space Filling Curves

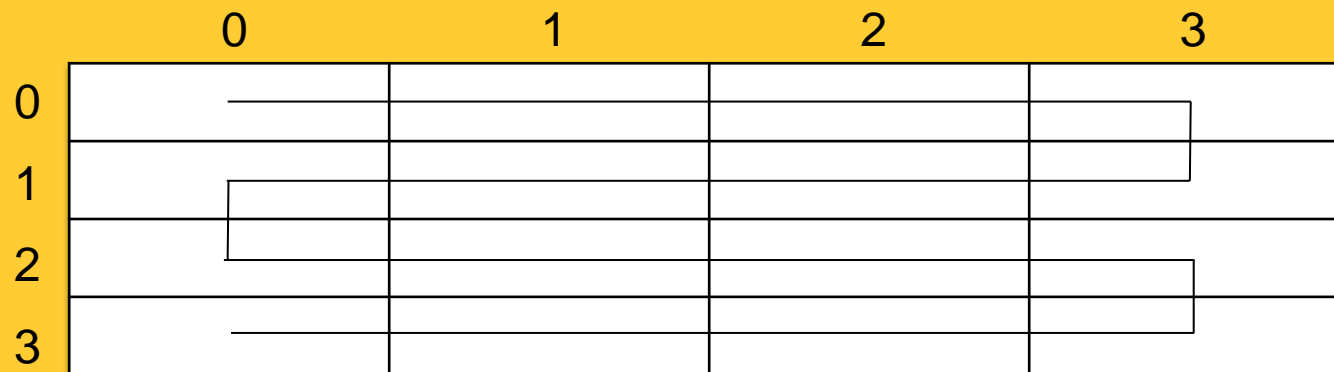
- Naïve:

	0	1	2	3
0	0	1	2	3
1	7	6	5	4
2	8	9	10	11
3	15	14	13	12



Space Filling Curves

- Naïve:



Space Filling Curves

- A better order:

	0	1	2	3
0	0	1	4	5
1	2	3	6	7
2	8	9	12	13
3	10	11	14	15



Space Filling Curves

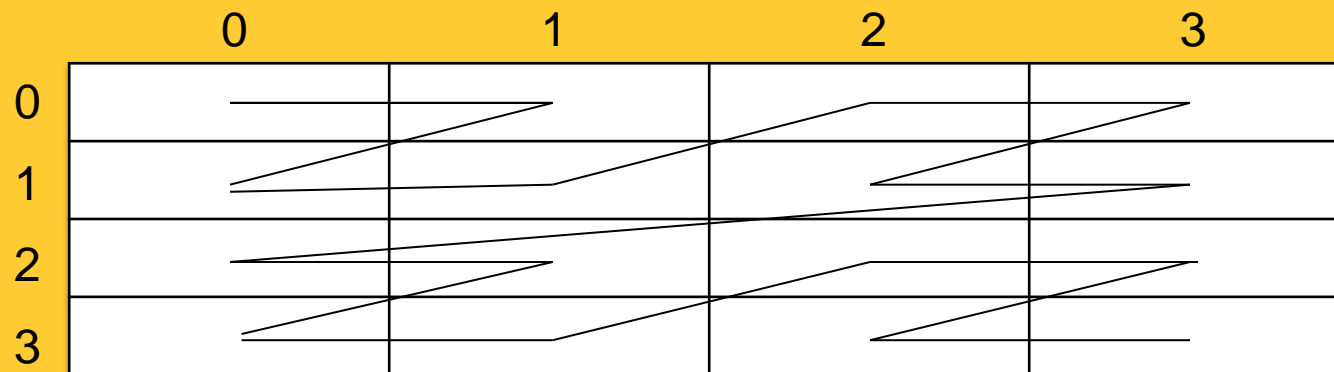
- A better order:

	0	1	2	3
0	0	1	4	5
1	2	3	6	7
2	8	9	12	13
3	10	11	14	15



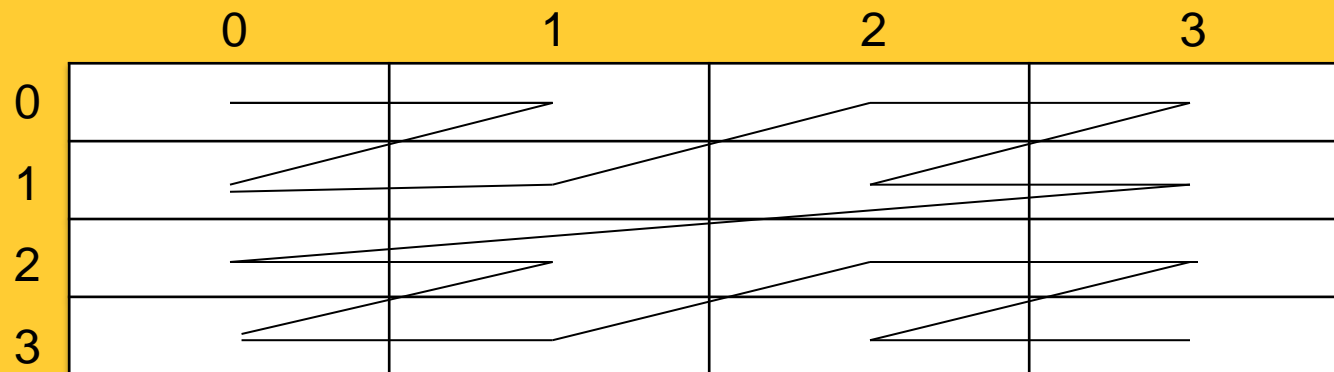
Space Filling Curves

- A better order:



Space Filling Curves

- A better order: Z-curve



Z - Curve

Figure 4.6

- Computing Z-value of a point
 - Generated from interleaving bits
 - x, y coordinate (see Fig. 4.6)
 - Connecting points by z-order
 - Looks like Ns or Zs
- Searching for a point P
 - Assume pointed sorted by Z-order
 - Compute Z-order for P
 - Binary search

x = 0 0 1 0
| | | |

y = 0 1 0 0 (2,4)
| | | |

n=0
□

n=1
□
Z

n=2

n=3

Figure 4.4



Space Filling Curves

- An alternative order: Hilbert curve

	0	1	2	3
0	0	1	14	15
1	3	2	13	12
2	4	7	8	11
3	5	6	9	10



Hilbert Curve

- More complex to generate
 - Due to rotations
- Searching for a point P
 - Assume ordered pointed
 - Compute Hilbert-order for P
 - Binary search

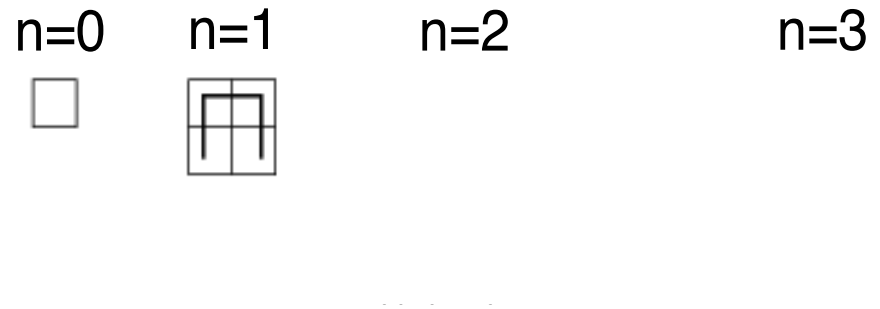
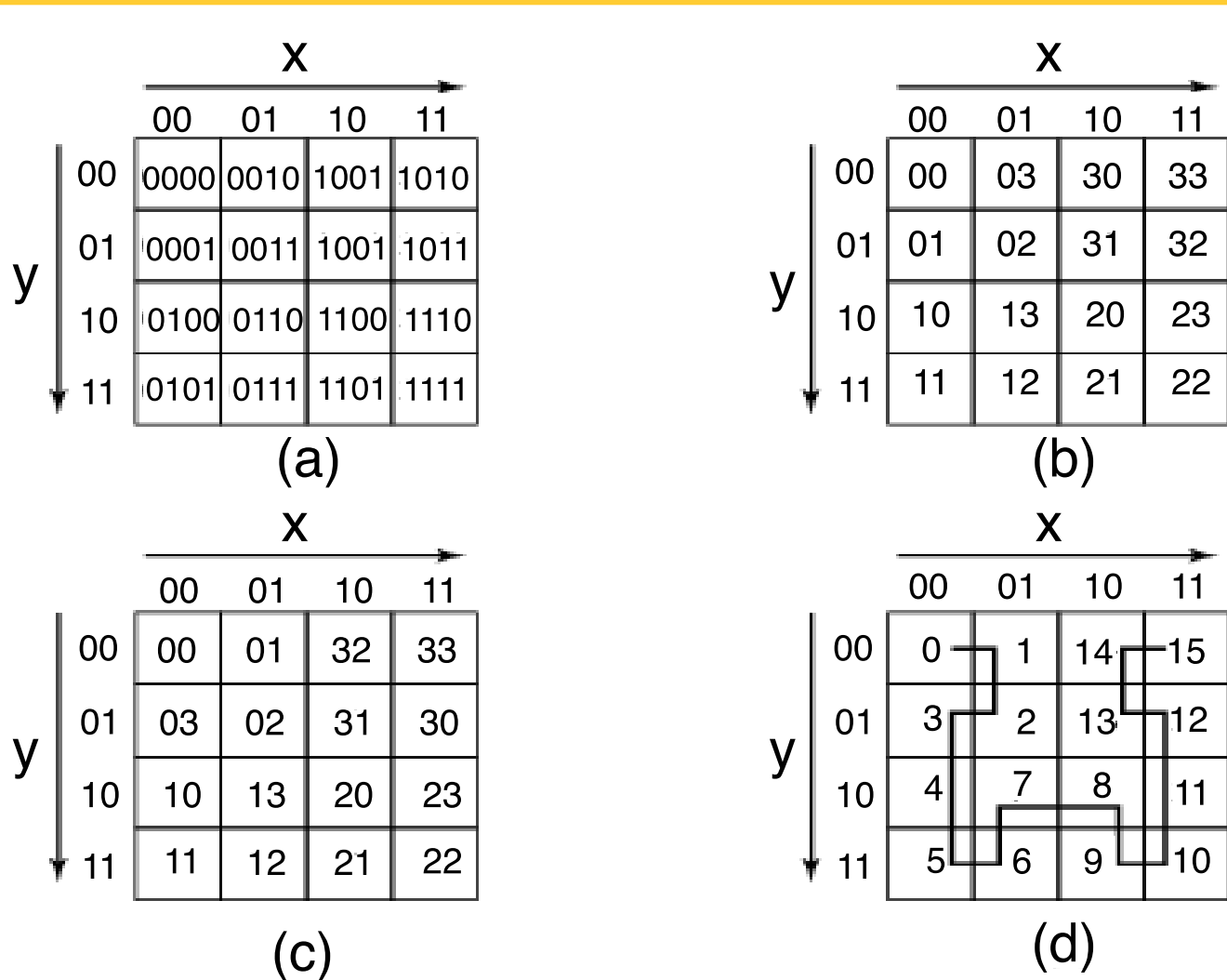


Figure 4.5

Calculating Hilbert Values (Optional Topic)

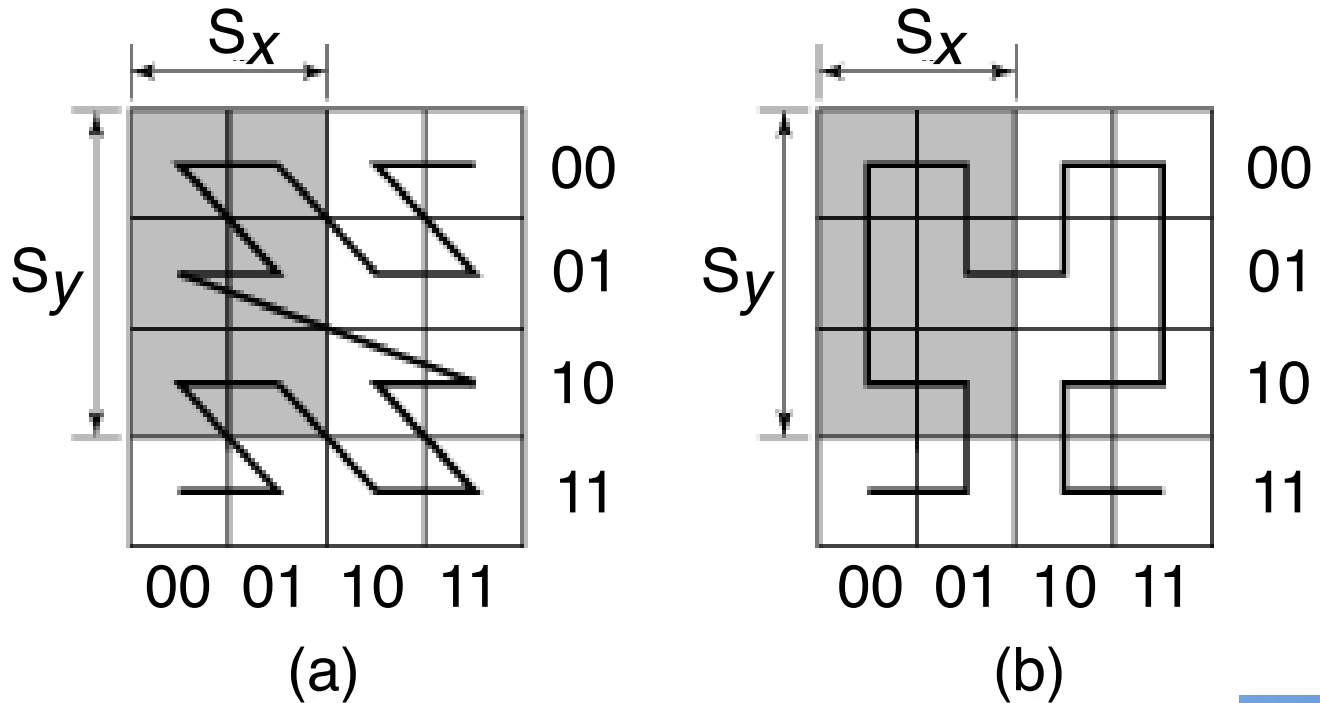
Figure 4.8



Handling Regions

- Challenge: non-unique values!

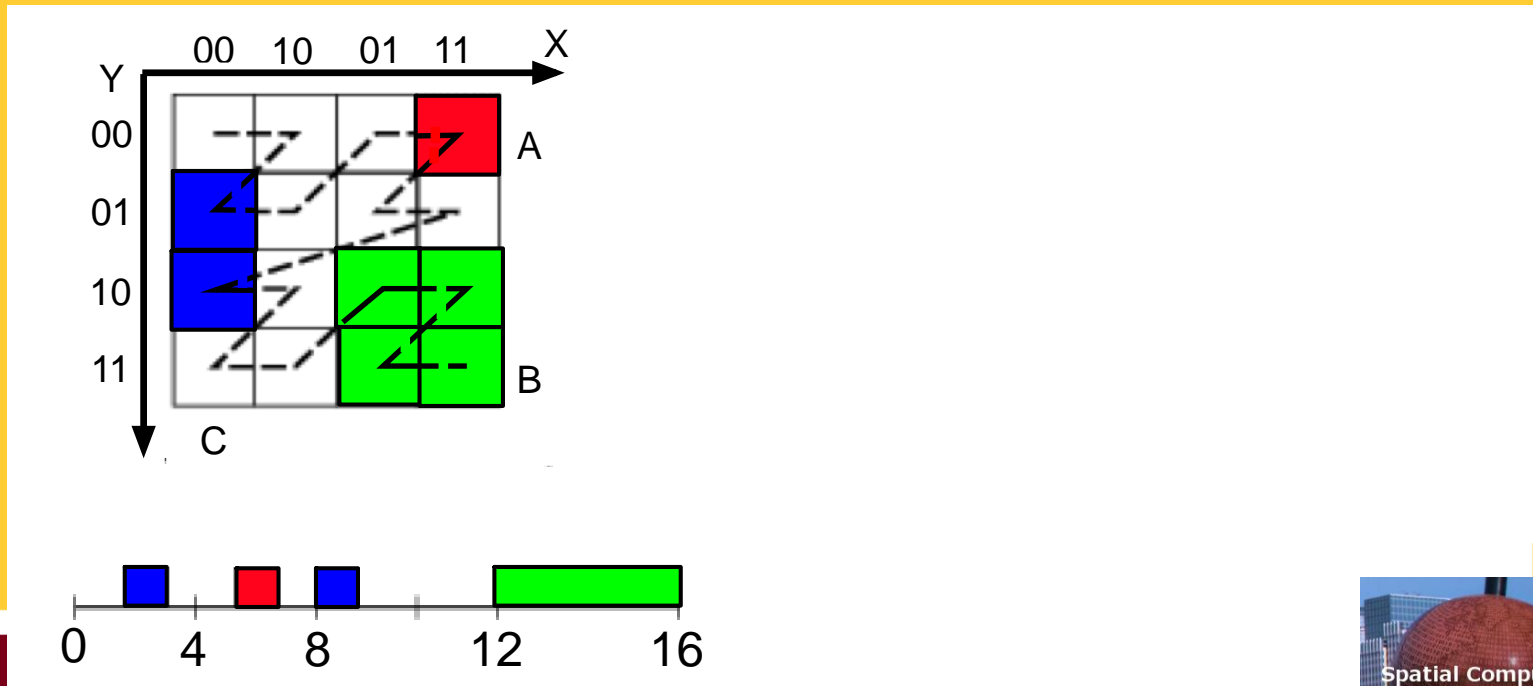
Figure 4.9



Z-order and Extended Objects

- Figure 4.7
 - Left part shows a map with spatial object A, B, C
 - Right part and Left bottom part Z-values within A, B and C
 - Note C gets z-values of 2 and 8, which are not close

Figure 4.7



Common assumptions for SDBMS physical model - Spatial Data

- Dimensionality of space is low, e.g. 2 or 3
 - Data types: OGIS data types
- Approximations for extended objects (e.g. linestrings, polygons)
 - Minimum Orthogonal Bounding Rectangle (MOBR or MBR)
 - $MBR(O)$ is the smallest axis-parallel rectangle enclosing an object O
- Supports filter and refine processing of queries



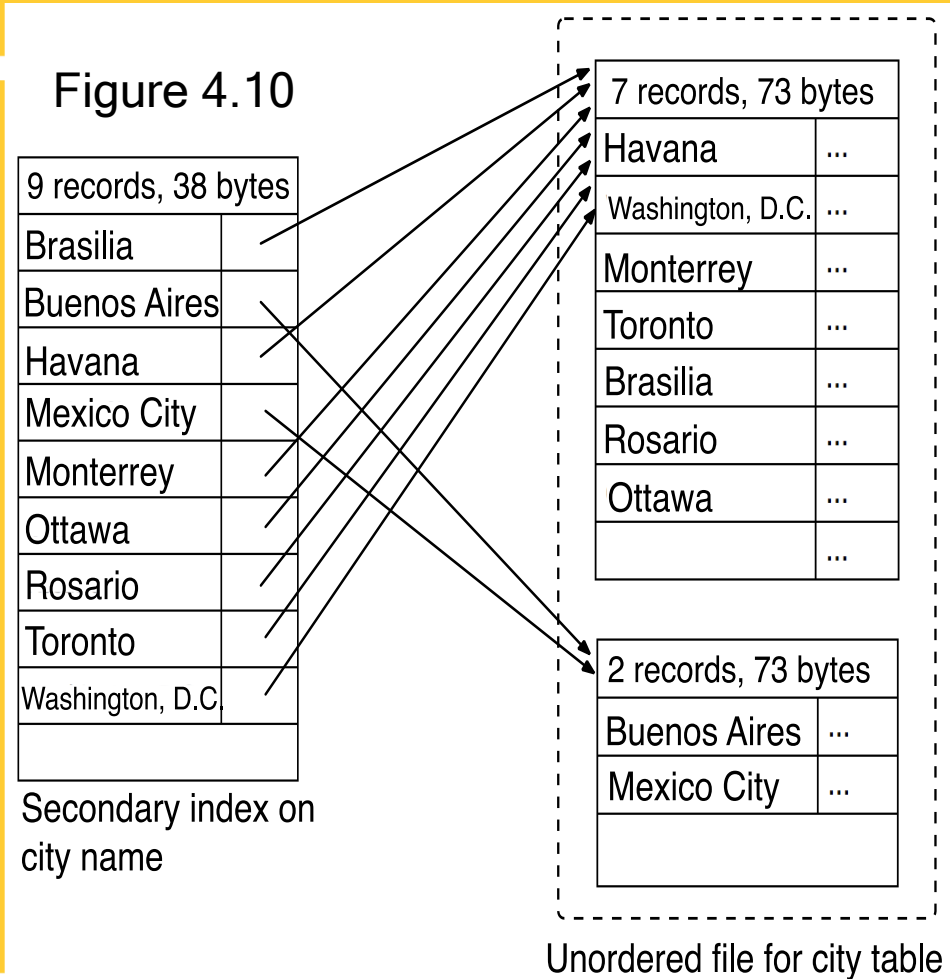
Common Spatial Queries

- Physical model provides simpler operations needed by spatial queries!
- Common Queries
 - Range query: Find all objects within a query rectangle.
 - k-nearest neighbor: Find k points closest to a query point.
 - Join query: Find all intersecting objects from two spatial datasets.



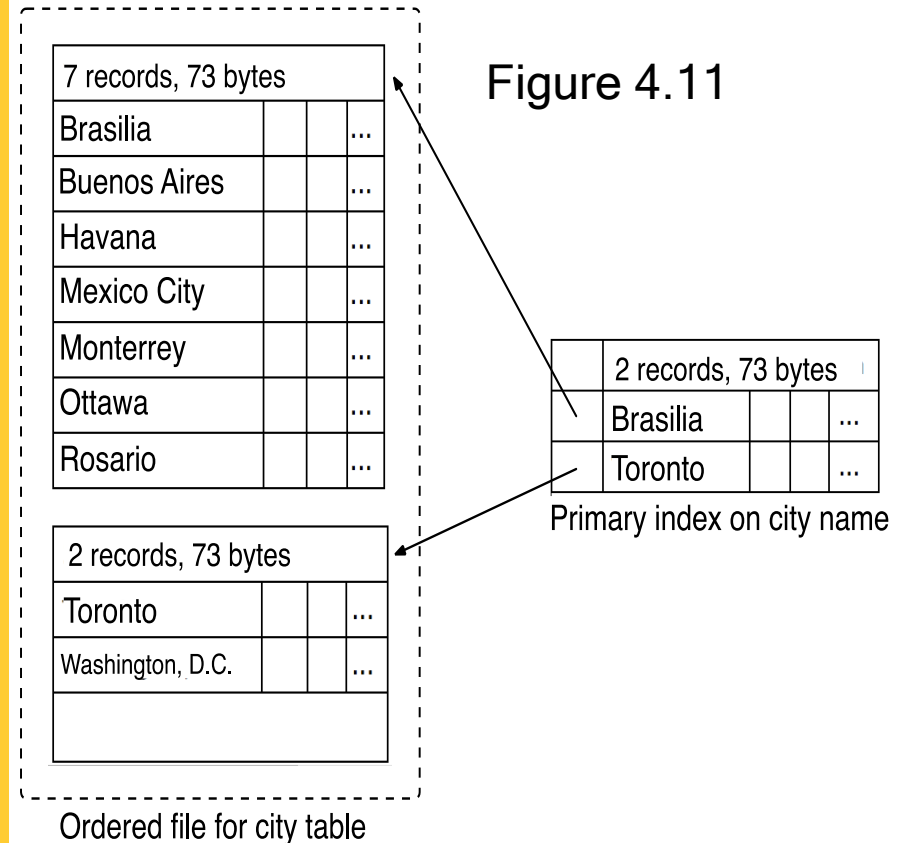
What is an index?

- Concept of an index
 - Auxiliary file to search a data file
 - Example: Fig. 4.10
- Index records have
 - Key value
 - Address of relevant data sector (see arrows in Fig. 4.10)
- Index records are ordered
 - Find, findnext, insert are fast
- Note assumption of total order
 - On values of indexed attributes



Classifying indexes

- Classification criteria
 - Data-file-structure
 - Key data type
 - Others
- Secondary index
 - Heap data file
 - 1 index record per data record
 - Example: Fig. 4.10
- Primary index
 - Data file ordered by indexed attribute
 - 1 index record per data sector
 - Example: Fig. 4.11



Question

Why is a file/table can have at most one primary index?



Question

Why is a file/table can have at most one primary index?

- (a) Reduce the ambiguity for users
- (b) Data file is physically ordered by the primary index
- (c) Save the storage space
- (d) Make query faster



Attribute Data Types and Indexes

- Index file structure depends on data type of indexed attribute
 - Attributes with total order
 - Example, numbers, points ordered by space filling curves
 - B-tree is a popular index organization
 - Spatial objects (e.g. polygons)
 - Spatial organization are more efficient
 - Hundreds of organizations are proposed in literature
 - Three main families are Grid Files, R-trees, and Quadtrees



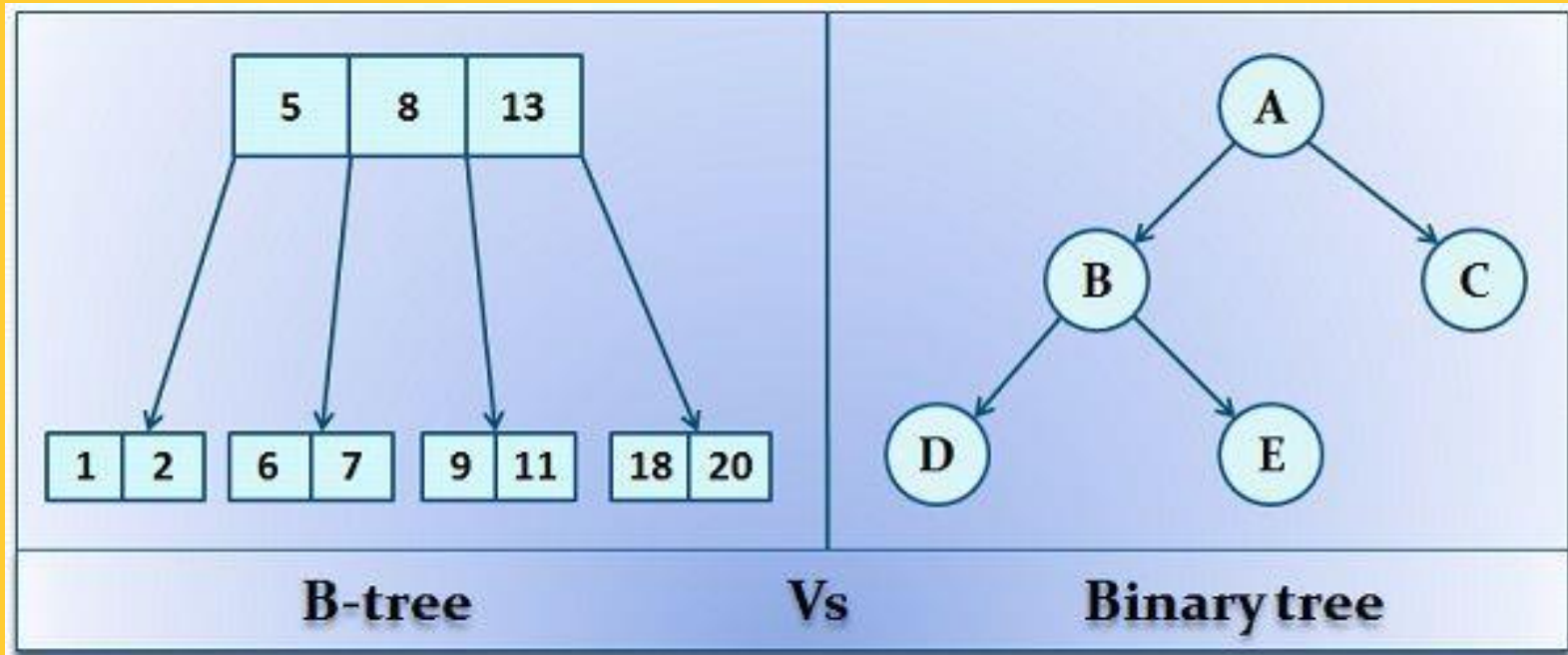
Popular Spatial Indexes

- Three main families are:
 - Spatial Grids (Grid Files)
 - R-tree (Data partitioning indexes)
 - Quadtree (Space partitioning indexes)



An Important Question

- Can I use B-tree for spatial indexing?



<https://techdifferences.com/difference-between-b-tree-and-binary-tree.html>

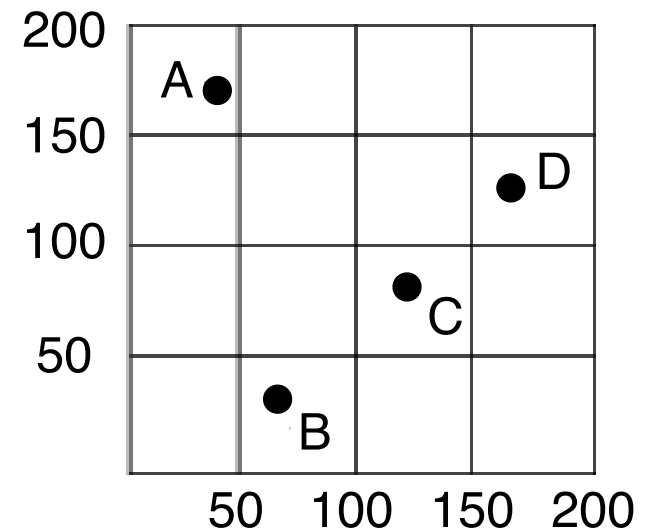
An Important Question

- Can I use B-tree for spatial indexing?
 - Yes, but it must have a total order
- For total order,
 - Either use lat or long dimensions
 - Or a space-filling curve order
- Or use two B-trees, one on lat, one on long

Spatial Grids

- Basic Idea: Divide geographic space
 - Example: latitude-longitude, ESRI Arc/SDE
 - Store data in each cell in distinct disk sector(s)
- Efficient for
 - uniformly distributed data
 - Operations: find, insert, nearest neighbor
- But may waste disk sectors
 - Empty cells
 - Non-uniform data distribution

Figure 4.12



From Grids to Grid Files

- 1. Use non-uniform grids (Fig. 4.14)
 - Linear scale store row and column boundaries

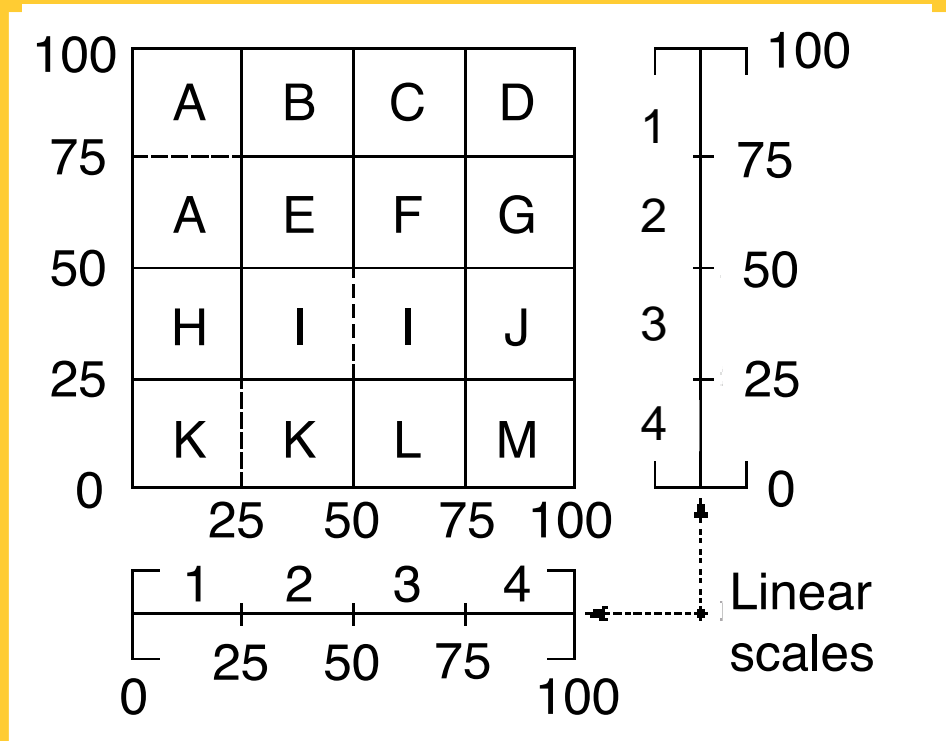
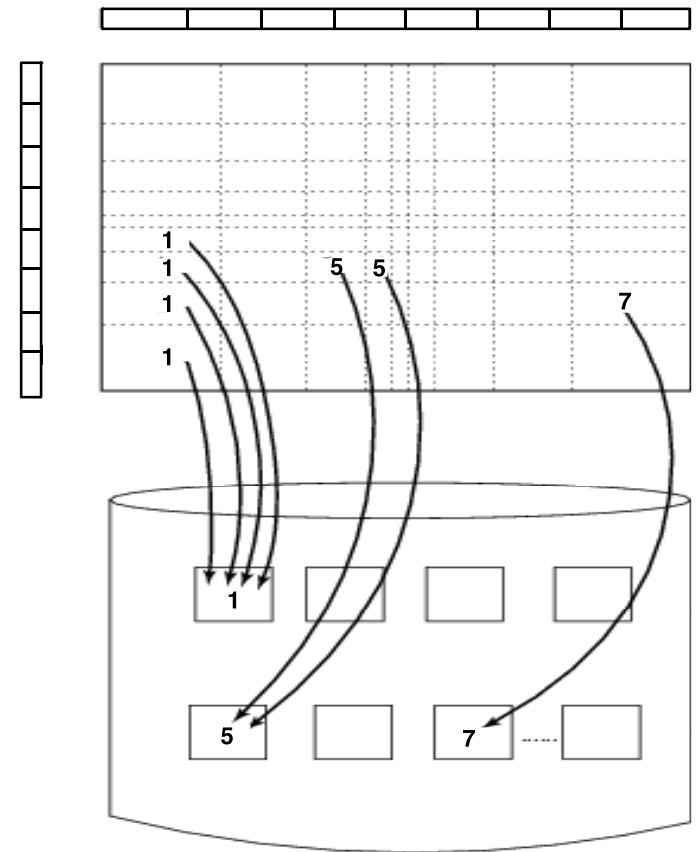


Figure 4.14

Grid Files - Search Operation

- Steps
 - Search linear scales
 - Identify selected grid directory cells
 - Retrieve selected disk sectors
- Optimization
 - Scales & grid directory in main memory
 - Efficient in terms of I/O costs
 - Needs large main memory for grid directory

Figure 4.13

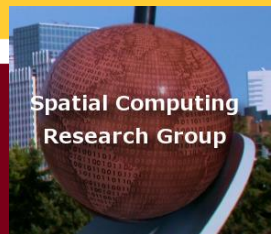
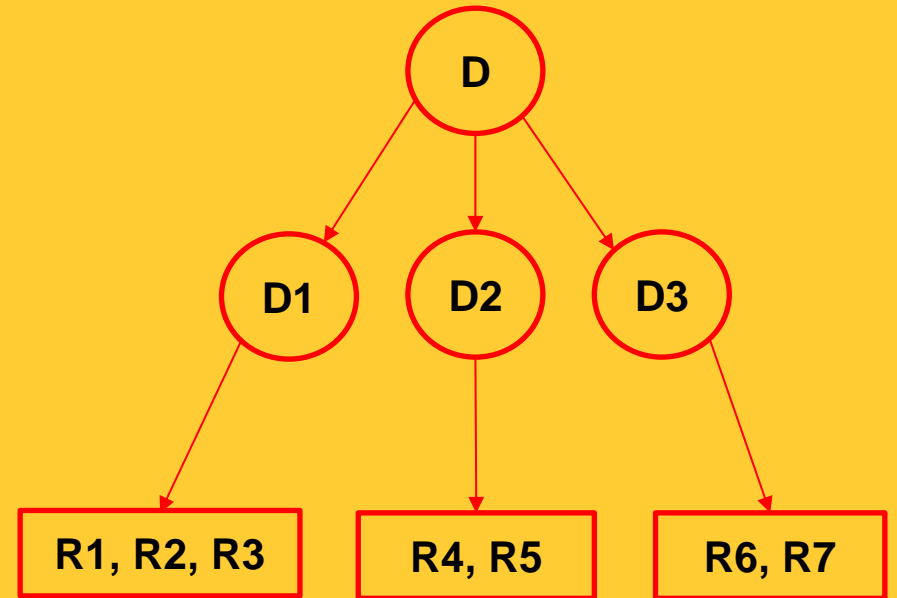
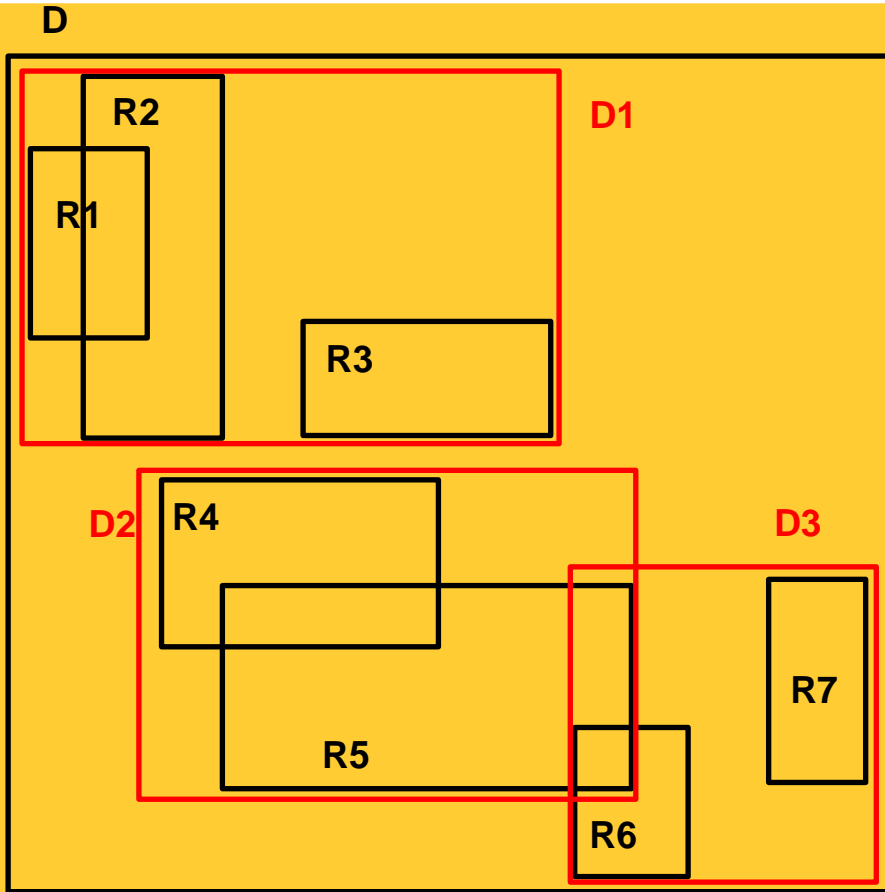


R-Tree Family - Basic Idea

- Hierarchical collection of rectangles organize spatial data
- Generalizes B-tree to spatial data sets

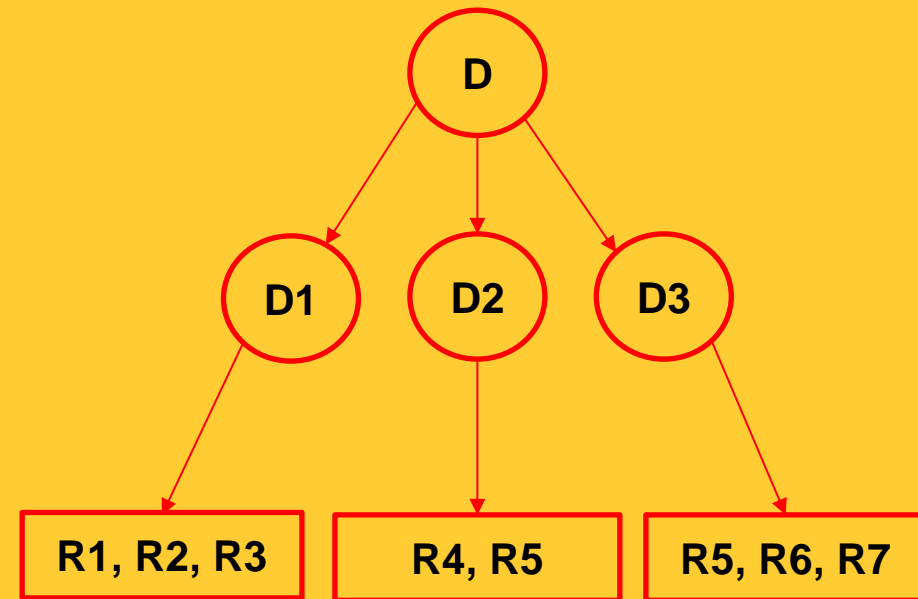
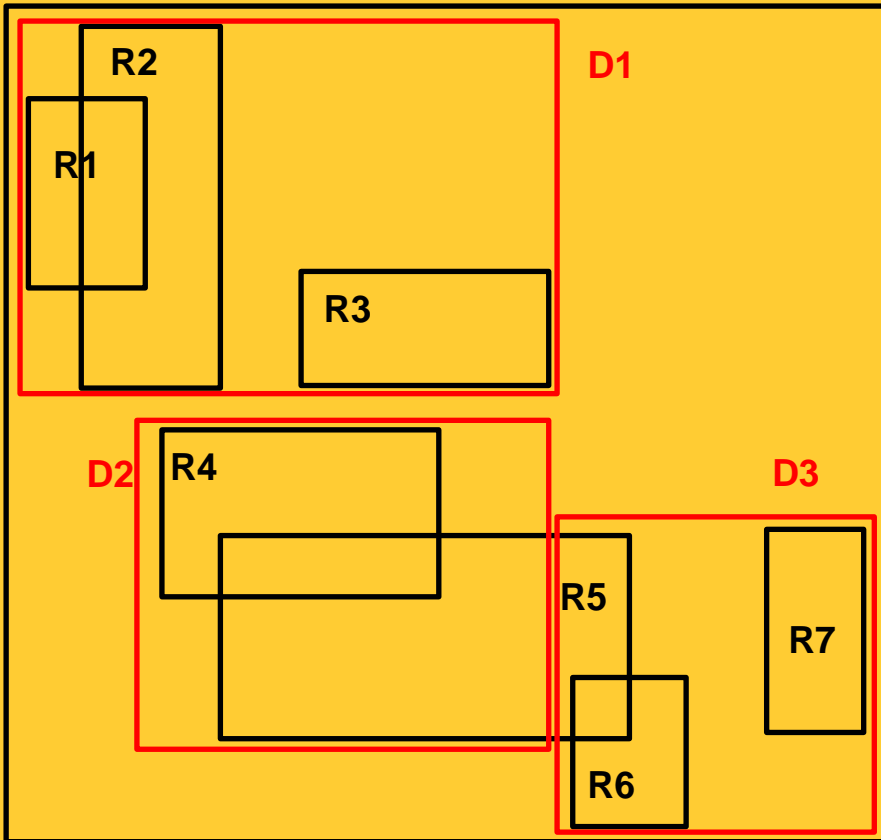


Example: R-tree



Example: R+ tree

D



R-Tree Family - Basic Idea

- Hierarchical collection of rectangles organize spatial data
- Generalizes B-tree to spatial data sets
- Properties of R-trees
 - Balanced
 - Nodes are rectangle
 - Child's rectangle within parent's
 - Possible overlap among rectangle!
 - Other properties in the section of R-tree family



Find Operation with R-tree

1. Search root to identify relevant children
2. Search selected children recursively

- Ex. Find record for rectangle 5

- Root search identifies child x
- Search of x identifies children b and c
- Search of b does not find object 5
- Search of c find object 5

Fig 4.16

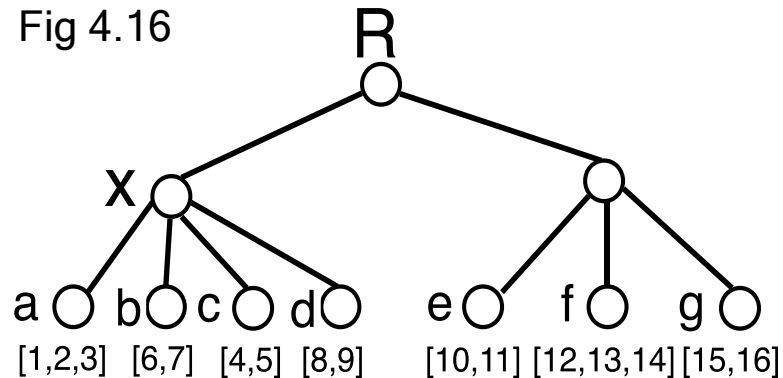
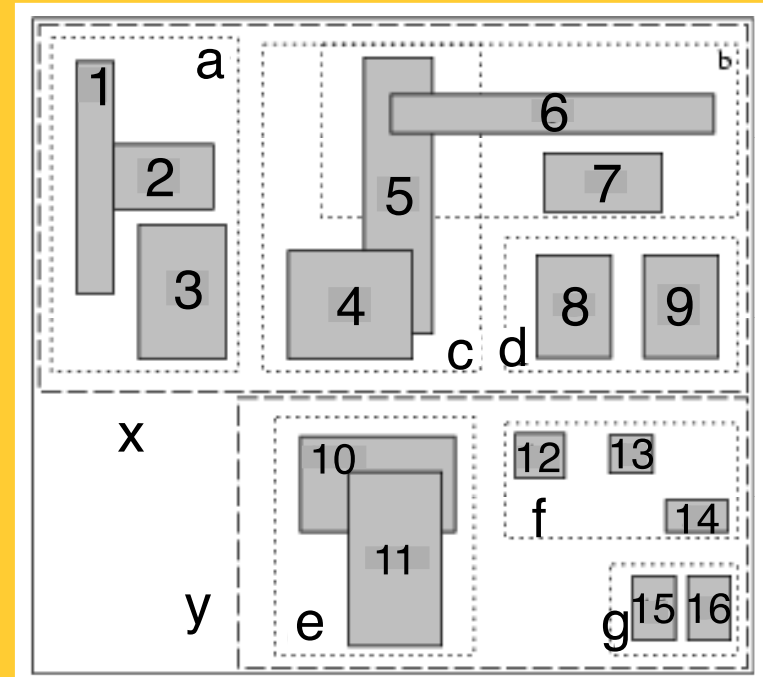
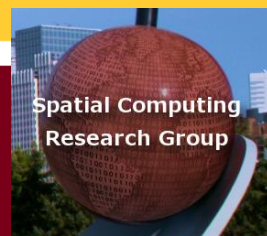


Fig 4.15



R-Tree Family - Classifying Members

- Handling of large spatial objects
 - Allow interior node rectangles to overlap - R-tree, R*-Tree
 - Duplicate objects but keep interior node rectangles disjoint - R+tree
- Selection of rectangles for interior nodes
 - Greedy procedures - R-tree, R+tree
 - Procedure to minimize coverage, overlap - R* tree
- Other criteria exist



R+tree

- Properties of R+trees
 - Balanced
 - Interior nodes are rectangle
 - Child's rectangle within parent's
 - Disjoint rectangles
 - Leaf nodes - MOBR of polygons or lines
 - Leaf's rectangle overlaps with parent's
 - Data objects may be duplicated across leafs
 - Other properties in the section of R-tree family
- Find operation - same as R-tree
 - But only one child is followed down in point queries

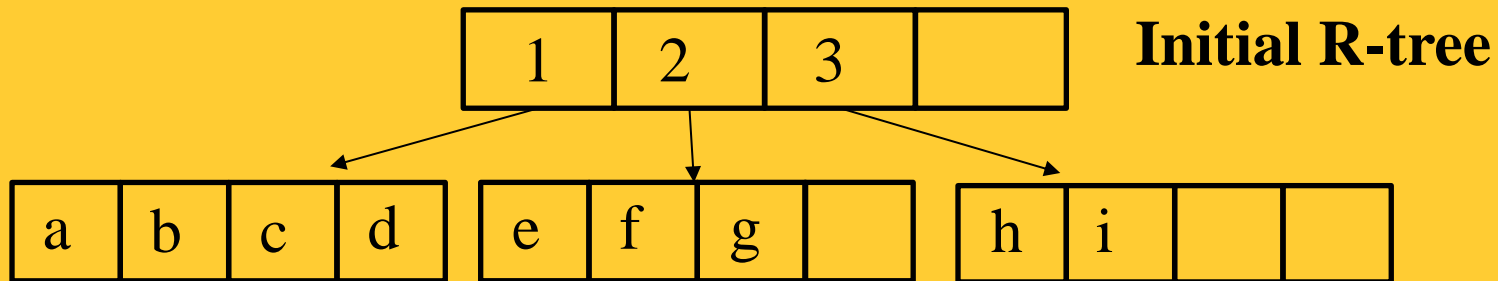
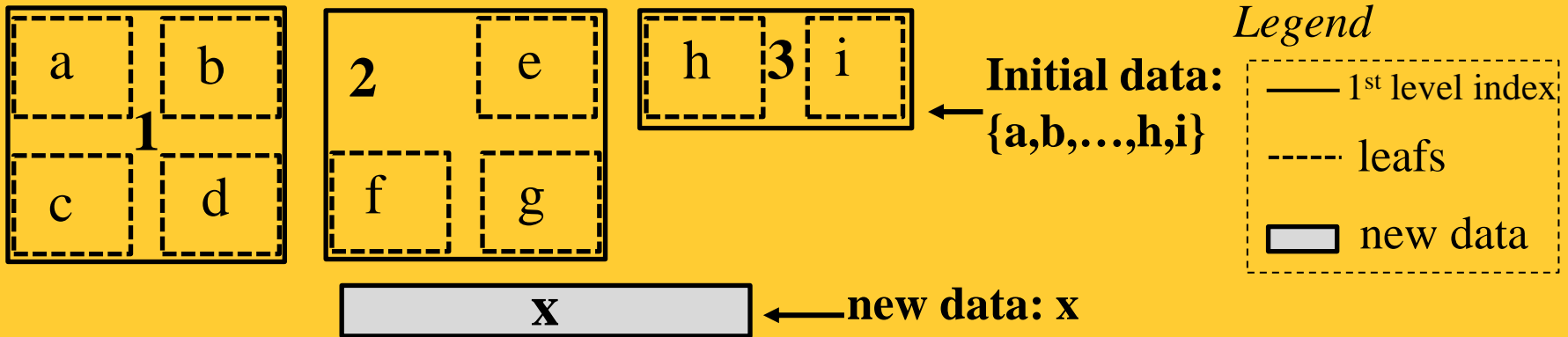


Comparison of Family of R trees

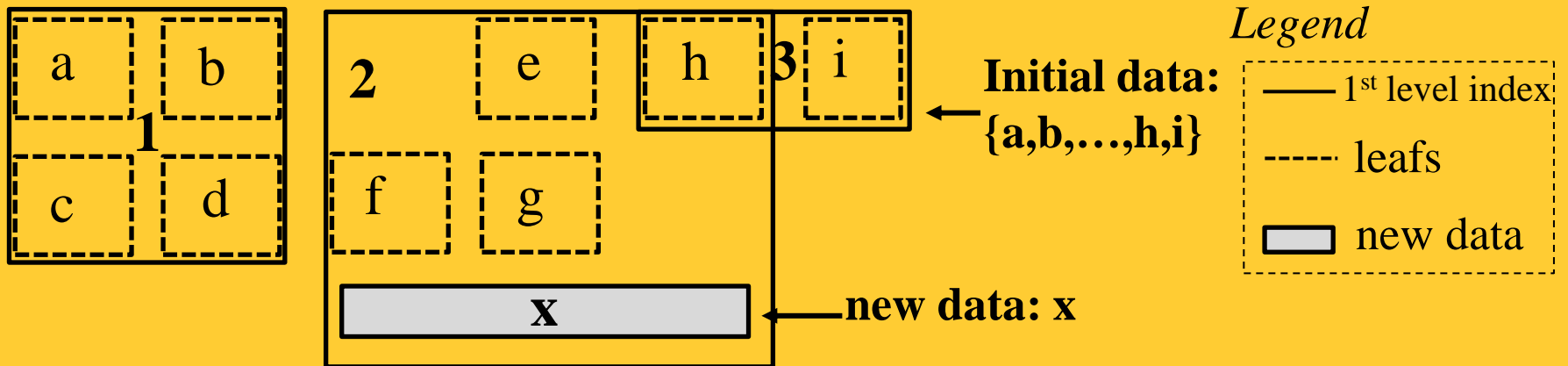
Property	R Tree	R + Tree	R * Tree
Overlap	Considerable	Allowed at leaf level	Minimum
Duplication	No	May be	No
Area	Minimizes	Does not consider	Minimizes
Margin	Not Applicable	Not Applicable	Minimizes
Reinsertion	No	No	Yes
Construction cost	Less	Somewhat More	Maximum



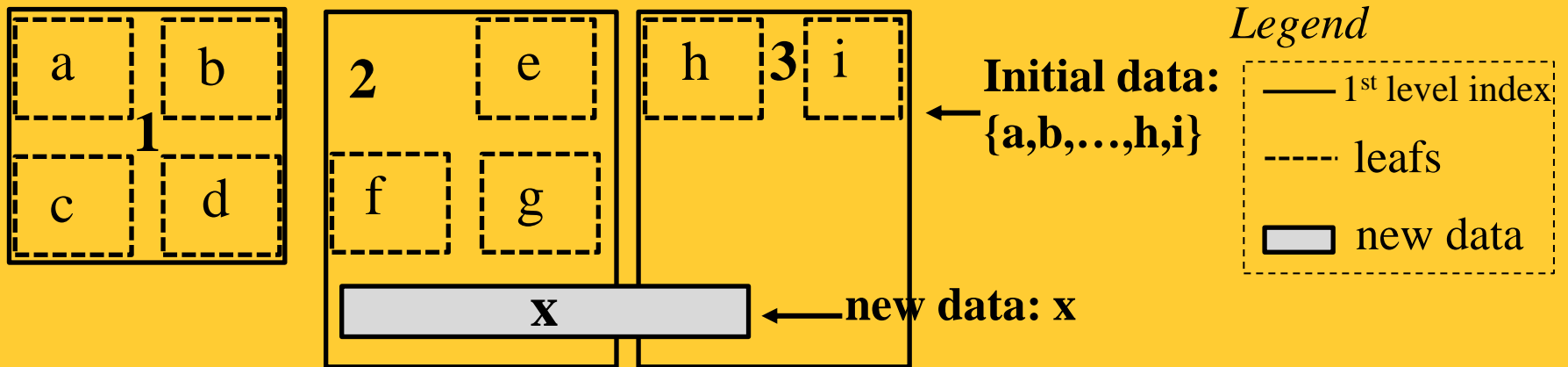
Dynamic Comparison (Insert)



Insert in R-Tree

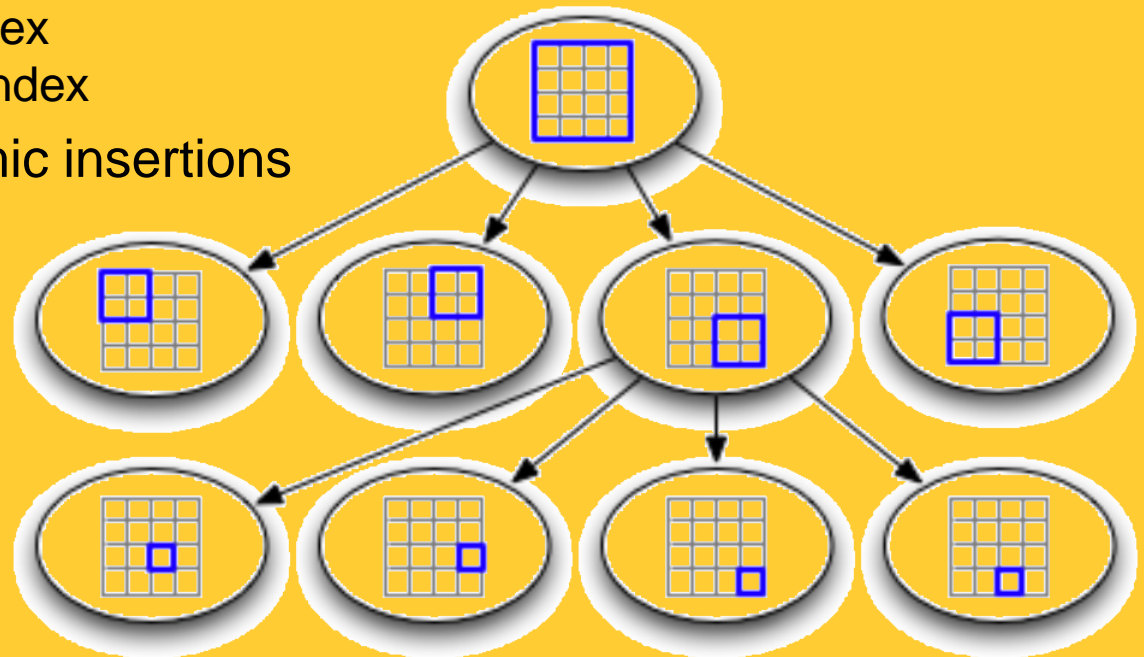


Insert in R+ Tree

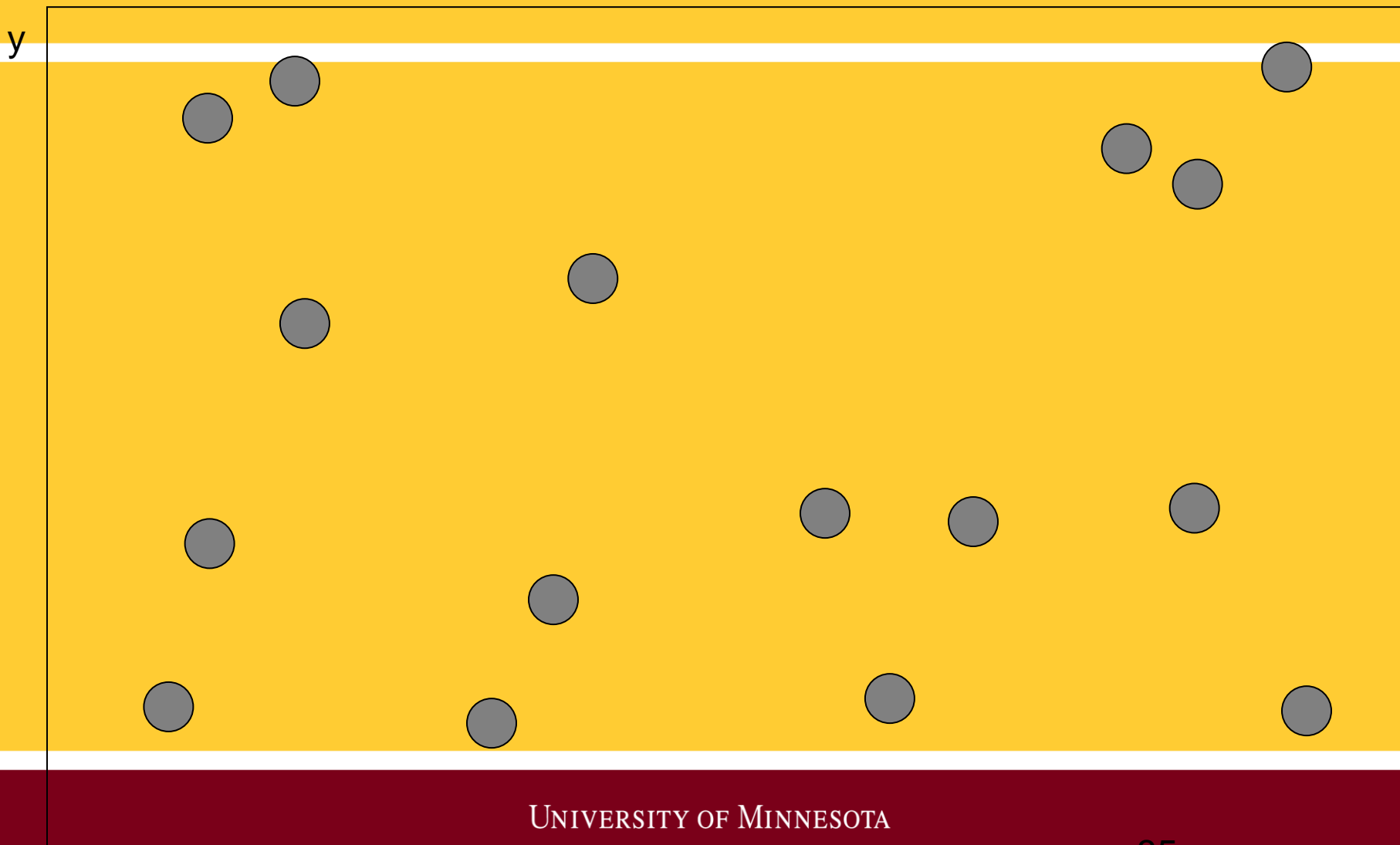


Quadtree

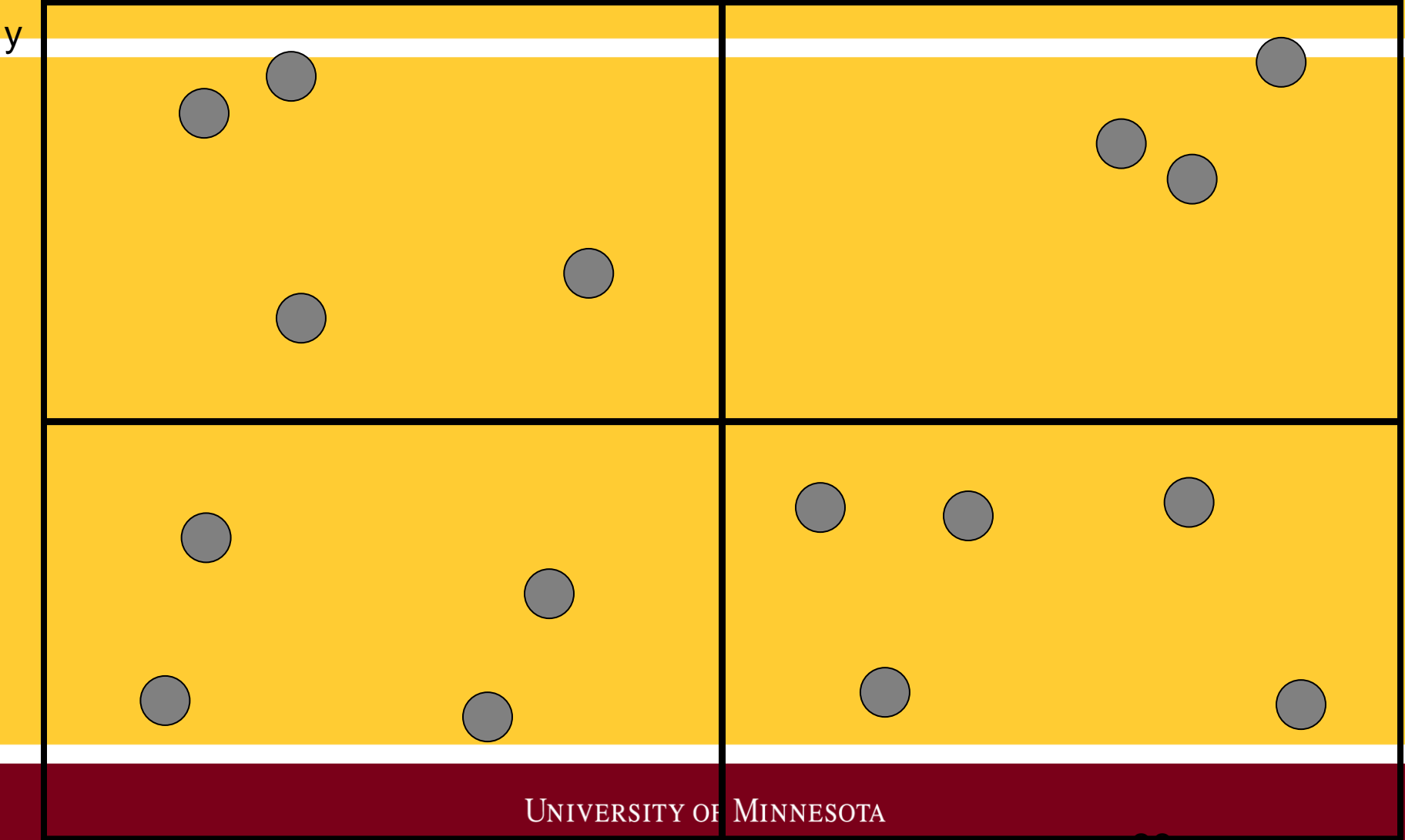
- Hierarchical index
- Partition boundaries depend only on parent's boundaries and not data
 - space-partitioning index
vs. data-partitioning index
- More efficient in dynamic insertions
- Memory-friendly



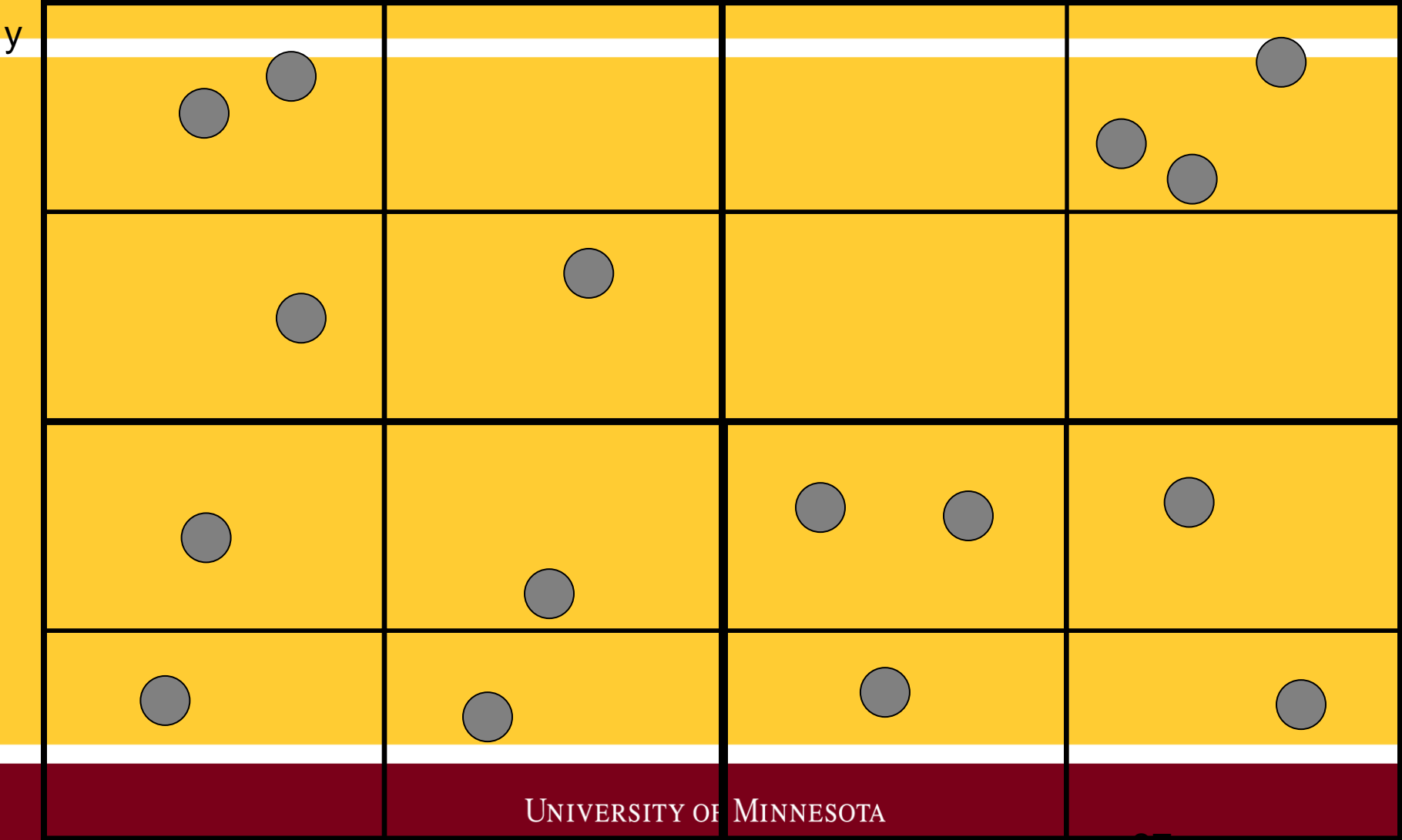
Quadtree: Example, Node capacity=2



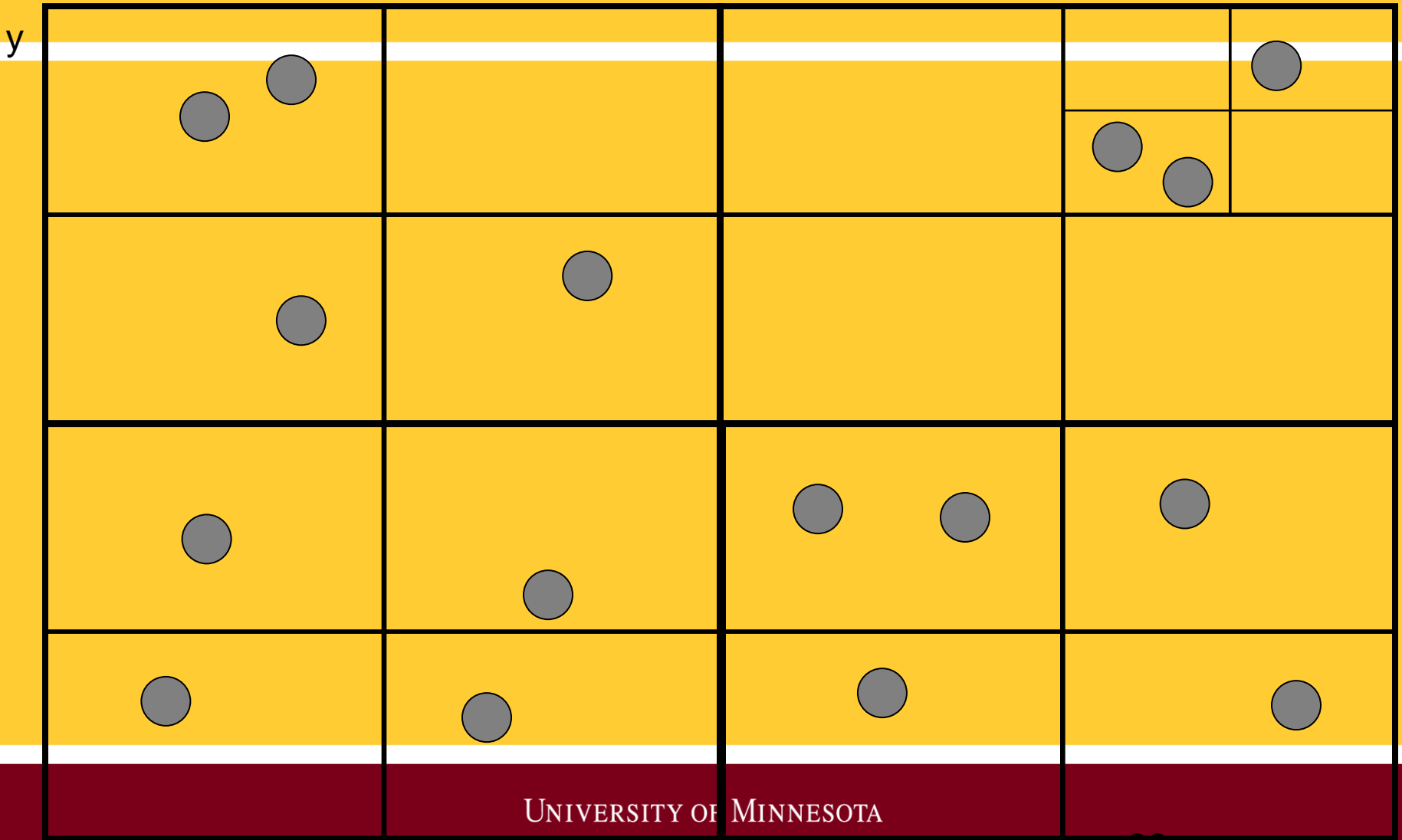
Quadtree: Example, Node capacity=2



Quadtree: Example, Node capacity=2

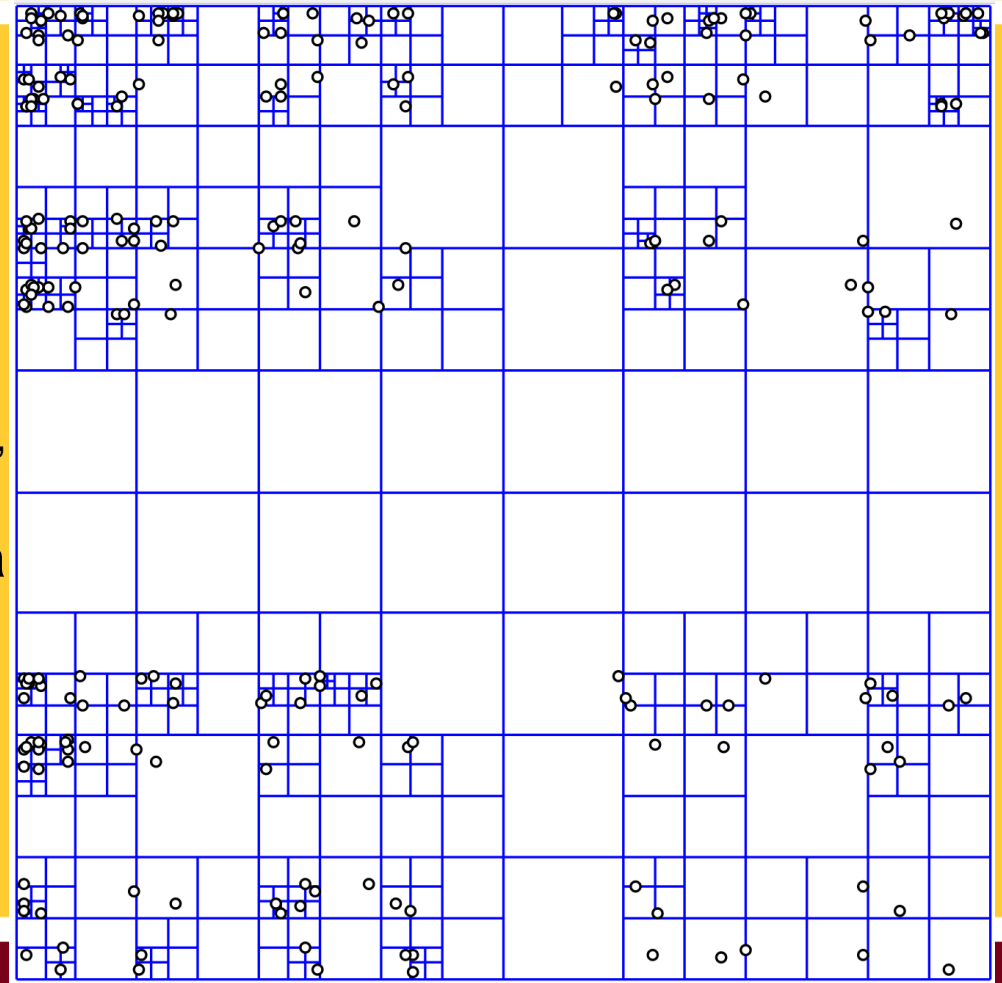


Quadtree: Example, Node capacity=2

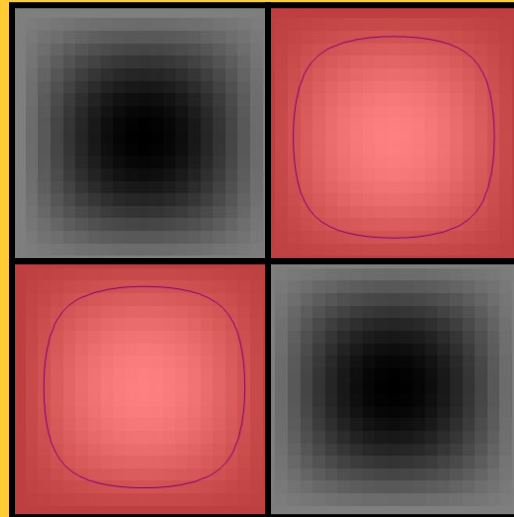




Quadtree

- Can be seen as a multi-level grid structure
- Different types:
 - Complete quadtree vs. partial quadtree
 - Region, point, point-region, edge, and polygon quad tree.
- Used in raster and vector data

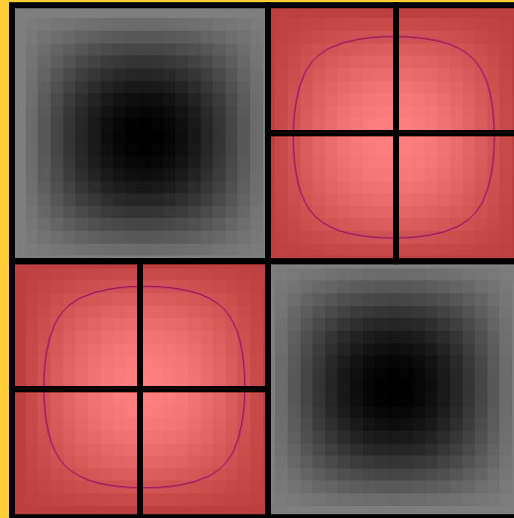




Quadtree: Example



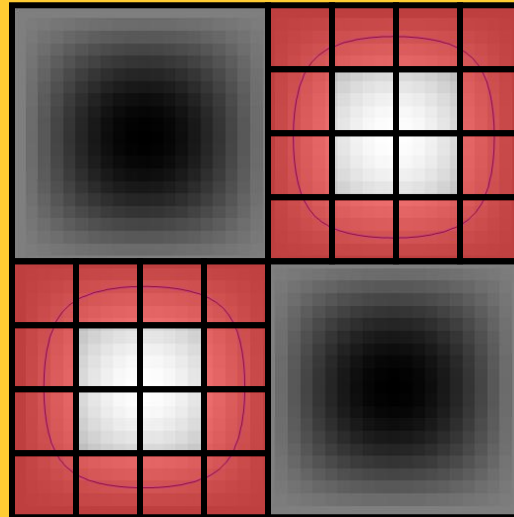
-  Iso-value not in {min,max}
-  Iso-value in {min,max}



Quadtree: Example



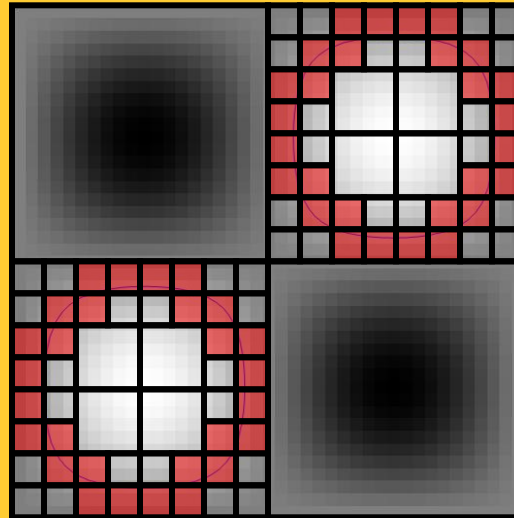
-  Iso-value not in {min,max}
-  Iso-value in {min,max}



Quadtree: Example



-  Iso-value not in $\{\min, \max\}$
-  Iso-value in $\{\min, \max\}$

Quadtree: Example



-  Iso-value not in $\{\min, \max\}$
-  Iso-value in $\{\min, \max\}$

Credits

- Prof. Tao Ju slides
 - http://www.cse.wustl.edu/~taoju/cse554/lectures/lect05_Contouring_II.ppt