Spatial Index Structures

Instructor: Cyrus Shahabi
Outline

• Grid File
• Z-ordering
• Hilbert Curve
• Quad Tree
  – PM
  – PR
• R Tree (next session)
  – R* Tree
  – R+ Tree
Grid File

- Hashing methods for multidimensional points (extension of Extensible hashing)
- Idea: Use a grid to partition the space → each cell is associated with one page
- Two disk access principle (exact match)
Grid File

• Start with one bucket for the whole space.
• Select dividers along each dimension.
Partition space into cells
• Dividers cut all the way.
Grid File

- Each cell corresponds to 1 disk page.
- Many cells can point to the same page.
- Cell directory potentially exponential in the number of dimensions
Grid File Implementation

• Dynamic structure using a grid directory
  – Grid array: a 2 dimensional array with pointers to buckets (this array can be large, disk resident) \( G(0, \ldots, nx-1, 0, \ldots, ny-1) \)
  – Linear scales: Two 1 dimensional arrays that used to access the grid array (main memory) \( X(0, \ldots, nx-1), Y(0, \ldots, ny-1) \)
Example

Grid Directory

Buckets/Disk Blocks

Linear scale X

Linear scale Y
Grid File Search

• Exact Match Search: at most 2 I/Os assuming linear scales fit in memory.
  – First use liner scales to determine the index into the cell directory
  – access the cell directory to retrieve the bucket address (may cause 1 I/O if cell directory does not fit in memory)
  – access the appropriate bucket (1 I/O)
  – E.g., \( X=(0, 1000, 1500, 1750, 1875, 2000) \); \( Y=(a, f, k, p, z) \)
    --- search for \([1980,w]\)

• Range Queries:
  – use linear scales to determine the index into the cell directory.
  – Access the cell directory to retrieve the bucket addresses of buckets to visit.
  – Access the buckets.
Grid File Insertions

- Determine the bucket into which insertion must occur.
- If space in bucket, insert.
- Else, split bucket
  - how to choose a good dimension to split?
  - ans: create convex regions for buckets.
- If bucket split causes a cell directory to split do so and adjust linear scales.
- insertion of these new entries potentially requires a complete reorganization of the cell directory---expensive!!!
Grid File Deletions

• Deletions may decrease the space utilization. Merge buckets

• We need to decide which cells to merge and a merging threshold

• Buddy system and neighbor system
  – A bucket can merge with only one *buddy* in each dimension
  – Merge adjacent regions if the result is a rectangle
Z-ordering

• Basic assumption: Finite precision in the representation of each coordinate, $K$ bits ($2^K$ values)
• The address space is a square (image) and represented as a $2^K \times 2^K$ array
• Each element is called a pixel
Z-ordering

• Impose a linear ordering on the pixels of the image → 1 dimensional problem

\[
Z_A = \text{shuffle}(x_A, y_A) = \text{shuffle}("01", "11")
= 0111 = (7)_{10}
\]
\[
Z_B = \text{shuffle}("01", "01") = 0011
\]
Z-ordering

- Given a point \((x, y)\) and the precision \(K\) find the pixel for the point and then compute the z-value
- Given a set of points, use a B+-tree to index the z-values
- A range (rectangular) query in 2-d is mapped to a set of ranges in 1-d
Queries

• Find the z-values that contained in the query and then the ranges

Q_A \rightarrow \text{range } [4, 7]
Q_B \rightarrow \text{ranges } [2,3] \text{ and } [8,9]
Hilbert Curve

- We want points that are close in 2d to be close in the 1d
- Note that in 2d there are 4 neighbors for each point where in 1d only 2.
- Z-curve has some “jumps” that we would like to avoid
- Hilbert curve avoids the jumps: recursive definition
Hilbert Curve- example

• It has been shown that in general Hilbert is better than the other space filling curves for retrieval *
• $H_i$ (order-$i$) Hilbert curve for $2^i \times 2^i$ array

* H. V. Jagadish: Linear Clustering of Objects with Multiple Attributes. ACM SIGMOD Conference 1990: 332-342
Quad Trees

• Region Quadtree
  – The blocks are required to be disjoint
  – Have standard sizes (squares whose sides are power of two)
  – At standard locations
  – Based on successive subdivision of image array into four equal-size quadrants
  – If the region does not cover the entire array, subdivide into quadrants, sub-quadrants, etc.
  – A variable resolution data structure
Example of Region Quadtree
PR Quadtree

• PR (Point-Region) quadtree
• Regular decomposition (similar to Region quadtree)
• Independent of the order in which data points are inserted into it
• 😞: if two points are very close, decomposition can be very deep
Example of PR Quadtree

Subdivide into quadrants until the two points are located in different regions
PM Quadtree

- PM (Polygonal-Map) quadtree family
  - PM1 quadtree, PM2 quadtree, PM3 quadtree, PMR quadtree, ... etc.
- PM1 quadtree
  - Based on regular decomposition of space
  - Vertex-based implementation
  - Criteria
    - At most one vertex can lie in a region represented by a quadtree leaf
    - If a region contains a vertex, it can contain no partial-edge that does not include that vertex
    - If a region contains no vertices, it can contain at most one partial-edge
PM Quadtree

PM1 quadtree

PM2 quadtree

PM3 quadtree
Example of PM1 Quadtree

- Each node in a PM quadtree is a collection of partial edges (and a vertex)
- Each point record has two field (x, y)
- Each partial edge has four field (starting_point, ending_point, left region, right region)
References

• National Technical University of Athens, Theoretical Computer Science II: Advanced Data Structures


• H. V. Jagadish: Linear Clustering of Objects with Multiple Attributes. ACM SIGMOD Conference 1990: 332-342