Spatial Index Structures

Cyrus Shahabi
Computer Science Department
University of Southern California
shahabi@usc.edu

Outline

- Introduction
- Spatial Indexing
- R-Tree
- $R^+$-Tree
- Quad Trees
Introduction

- **Spatial objects**
  - Points, lines, rectangles, regions, ...
- **Hierarchical data structures**
  - Based on recursive decomposition, similar to divide and conquer method, like B-tree.
- **Why not B-Tree?**
  - More than one dimension
  - Concept of closeness relies on all the dimensions of the spatial data
- **Spatial index structures demo**

Spatial Indexing

- **Mapping spatial object into point**
  - In either same, lower, or higher dimensional spaces
  - Good for storage purposes
  - Problems with queries like finding the nearest objects
- **Bucketing methods**
  - Based on spatial occupancy
  - Decomposing the space from which the data is drawn
    - Minimum bounding rectangle (MBR): e.g., R-Tree
    - Disjoint cells: e.g., R*-Tree
    - Blocks of uniform size
    - Distribution of the data: e.g., quadtree
    - data-dependent
    - data-independent
R-Tree
[proposed in 1984 by Guttman]

Based on Minimum Bounding Rectangle
(m,M)=(1,3)

- bounding rectangles could overlap each others (e.g., R3 vs R4)
- an object is only associated with one bounding rectangle

R-Tree
[proposed in 1984 by Guttman]

- Height-balanced tree similar to B-tree for k-dimensions
- Every leaf node contains between m (m ≤ M/2) and M index records, unless it is the root
- For each index record (I, tuple-identifier) in a leaf node, I is the MBR that contains the n-dimensional data object represented by the indicated tuple
- Every non-leaf node has between m and M children unless it is the root
- For each entry (I, child-pointer) in a non-leaf node, I is the MBR that spatially contains the rectangles in the child node.
- All leaves appear on the same level
- The root node has at least two children unless it is a leaf
**Insertion Processes**

- A new index entry
  - ChooseLeaf
  - Has room
    - Yes: Install X
    - No: SplitNode
  - Adjust all related entries
  - AdjustTree

**Processes of Quadratic Split**

* (page 52 in Guttman's paper)

- Pick first entry for each group
- Run PickSeeds
- Different variant:
  - Exhaustive
  - Quadratic
  - Linear
  - Packed
  - Hilbert Packed
  - …etc.
Processes of Quadratic Split
(page 52 in Guttman’s paper)

PickSeeds
PS1 [Calculate inefficiency of grouping entries together]
For each pair of E1 and E2, compose a rectangle R including E1 and E2
Calculate \( d = \text{area}(R) - \text{area}(E1) - \text{area}(E2) \)
PS2 [Choose the most wasteful pair]
Choose the pair with the largest \( d \)

Check if done
Select entry to assign
(PickNext)
Processes of Quadratic Split
(page 52 in Guttman's paper)

Pick first entry for each group (PickSeeds)

- G1
- G2
- l
- m

Check if done

No

Select entry to assign (PickNext)

- G1
- G2
- l
- k

Processes of Quadratic Split
(page 52 in Guttman's paper)

Pick first entry for each group (PickSeeds)

- G1
- G2
- l
- m

Check if done

No

Select entry to assign (PickNext)

- G1
- G2
- l
- k
Processes of Quadratic Spilt
(page 52 in Guttman's paper)

Pick first entry for each group
(PickSeeds)

G1 | G2
---|---
1 | m

Check if done

No

Select entry to assign
(PickNext)

G1 | G2
---|---
1 | m
k | X

Processes of Quadratic Spilt
(page 52 in Guttman's paper)

Pick first entry for each group
(PickSeeds)

G1 | G2
---|---
1 | m

Check if done

No

Select entry to assign
(PickNext)

G1 | G2
---|---
1 | m
k | X
Processes of Quadratic Spilt
(page 52 in Guttman's paper)

Pick first entry for each group
(PickSeeds)

Check if done

No

Select entry to assign
(PickNext)

Pick (m,M)=(2,4)
Build a R-Tree for these spatial data
Hint: You could use the Spatial index structures demo application step by step
Search Object in R-Tree

Input \((T, j)\)

Leaf?

Yes

Check each entry \(E\) of \(T\)

Overlap

Yes

Qualifying record

No

Search each subtree \(E\) of \(T\)

Overlap

Yes

Main Drawbacks of R-Tree

- R-tree is not unique, rectangles depend on how objects are inserted and deleted from the tree.
- In order to search some object you might have to go through several rectangles or the whole database
  - Why?
  - Solution?
R+*-Tree

- Overcome problems with R-Tree
- If node overlaps with several rectangles insert the node in all
- Decompose the space into disjoint cells

R+-Tree Properties

- R+-tree and cell-trees used approach of decomposing space into cells
  - R+-trees deals with collection of objects bounded by rectangles
  - Cell tree deals with collection of objects bounded by convex polyhedron
- R+-trees is extension of k-d-B-tree
- Retrieval times are smaller
- When summing the objects, needs eliminate duplicates
- Again, it is data-dependent
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
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)

Question?
B-Tree: Definition

A B-Tree of order $M$ is a height-balanced tree

1. All leaves are on the same level
2. All nodes have at most $M$ children
3. All internal nodes except the root have at least $M/2$ children

B-Tree: Insertion

A new index entry

ChooseLeaf

Has room

Yes
Install $X$

No
SplitNode

AdjustTree