Spatial Index Structures
(R-tree Family)

Instructor: Cyrus Shahabi
Problem

• Given a collection of geometric objects (points, lines, polygons, ...)
• organize them on disk, to answer spatial queries (range, nn, etc)
R-trees

• [Guttman 84] Main idea: extend B+-tree to multi-dimensional spaces!

  — (only deal with Minimum Bounding Rectangles - MBRs)
R-trees

• A multi-way external memory tree
• Index nodes and data (leaf) nodes
• All leaf nodes appear on the same level
• Every node contains between $m$ and $M$ entries
• The root node has at least 2 entries (children)
Example

• eg., w/ fanout 4: group nearby rectangles to parent MBRs; each group -> disk page
Example

- F=4
Example

- $F = 4$
R-trees - format of nodes

- \{(MBR; obj_ptr)\} for leaf nodes
R-trees - format of nodes

- \{(MBR; node_ptr)\} for non-leaf nodes

| x-low; x-high | y-low; y-high | node ptr | ...
|--------------|-------------|----------|---

A B C

P1 P2 P3 P4
Insertion Processes

A new index entry

ChooseLeaf

Has room

Yes

Install X

No

SplitNode

Adjust all related entries

AdjustTree

Different variant:
- Exhaustive
- Quadratic
- Linear
- Packed
- Hilbert Packed
- …etc.
Processes of Quadratic Spilt
(page 52 in Guttman’s paper [1])

Pick first entry for each group
Run PickSeeds
**Processes of Quadratic Spilt**

*(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 \( d \)
Processes of Quadratic Split
(page 52 in Guttman’s paper)

Pick first entry for each group
(PickSeeds)

<table>
<thead>
<tr>
<th>G1</th>
<th>G2</th>
</tr>
</thead>
<tbody>
<tr>
<td>l</td>
<td>m</td>
</tr>
</tbody>
</table>

Check if done

No

Select entry to assign
(PickNext)
Processes of Quadratic Spilt
(page 52 in Guttman’s paper)

Pick first entry for each group (PickSeeds)

<table>
<thead>
<tr>
<th>G1</th>
<th>G2</th>
</tr>
</thead>
<tbody>
<tr>
<td>l</td>
<td>m</td>
</tr>
</tbody>
</table>

Check if done

No

PickNext
PN1 [Determine cost of putting each entry in each group] For each entry E
calculate $d_1$ = the increased MBR area required for G1
calculate $d_2$ = the increased MBR area required for G2

PN2 [Find entry with greatest preference for one group] Choose the entry with the maximum difference between $d_1$ and $d_2$
Processes of Quadratic Spilt

(page 52 in Guttman’s paper)

Pick first entry for each group (PickSeeds)

Check if done

Select entry to assign (PickNext)

G1  G2
l   m

G1  G2
l   m
k
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)
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)
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)
Exercise

- \((m,M)=(2,4)\)

- 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

E=j

Yes

Qualifying record

No

For each subtree E of T

(E,j)

Yes

(E Contains j)

Yes

(E= j)

No

For each subtree E of T

(E Contains j)
Overlap query in R-Tree
(find objects that overlap with Z)

Input (T, Z)

Leaf?

Yes

Check each entry E of T

Overlap

Yes

Qualifying record

No

Search each subtree E of T

Overlap

Yes

Overlap (E,j)

Yes

Qualifying record
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
R-tree

• The original R-tree tries to minimize the area of each enclosing rectangle in the index nodes.

• Is there any other property that can be optimized?

R*-tree \rightarrow Yes!
R*-tree

• Optimization Criteria:
  – (O1) Area covered by an index MBR
  – (O2) Overlap between index MBRs
  – (O3) Margin (perimeter) of an index rectangle
  – (O4) Storage utilization

• Sometimes it is impossible to optimize all the above criteria at the same time!
R*-tree

• ChooseSubtree:
  – If next node is a leaf node, choose the node using the following criteria:
    • Least overlap enlargement
    • Least area enlargement
    • Smaller area
  – Else
    • Least area enlargement
    • Smaller area
R*-tree

- **SplitNode**
  - Choose the axis to split
  - Choose the two groups along the chosen axis

- **ChooseSplitAxis**
  - Along each axis, sort rectangles and break them into two groups (M-2m+2 possible ways where one group contains at least m rectangles). Compute the sum S of all margin-values (perimeters) of each pair of groups. Choose the one that minimizes S

- **ChooseSplitIndex**
  - Along the chosen axis, choose the grouping that gives the minimum overlap-value
R*-tree

• Forced Reinsert:
  – defer splits, by forced-reinsert, i.e.: instead of splitting, temporarily delete some entries, shrink overflowing MBR, and re-insert those entries

• Which ones to re-insert?
• How many? A: 30%
References

- Norbert Beckmann, et al., The R*-tree: an efficient and robust access method for points and rectangles, SIGMOD 1990
- Roussopoulos et al., The R+-Tree: A Dynamic Index for Multi-Dimensional Objects, VLDB 1987
- National Technical University of Athens, Theoretical Computer Science II: Advanced Data Structures