Module 5: Disk-based Index Structures

Lecture 22: SS-trees,SR-trees,P-trees

The Lecture Contains:

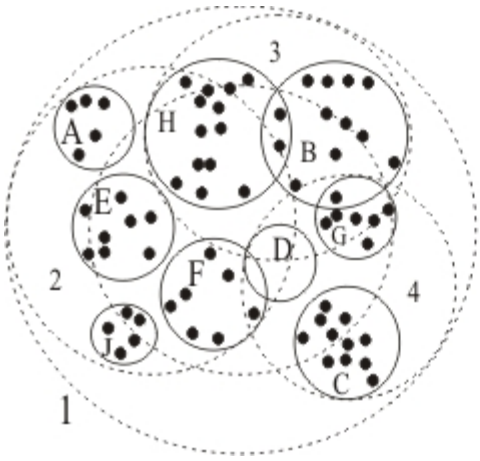
- SS-tree
  - Bounding spheres and bounding rectangles
- SR-tree
  - Structure and algorithms
  - Performance
- P-tree
  - Search performance and overlap
  - Effect of dimensionality on overlap

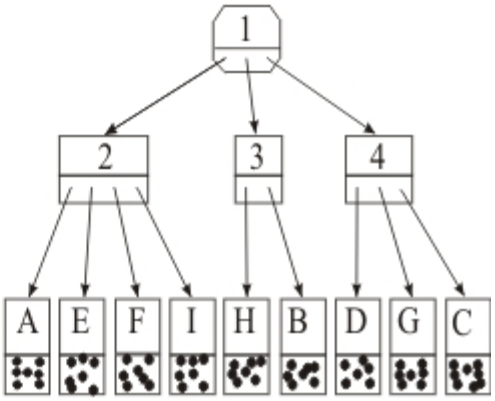
SS-tree

- Uses bounding spheres instead of bounding rectangles

SS-tree

- Uses bounding spheres instead of bounding rectangles
- Motivated by range and kNN queries which are hyper-spheres
- Center of sphere is centroid of points
- Maintains total number of points in the subtree
- Higher fanout due to smaller storage requirements
- Considerable volume overlap
- Split axis is chosen based on variance
- Better than R*-trees





Bounding spheres and bounding rectangles

- SS-tree has regions with shorter diameters
- R*-tree has regions with smaller volumes
- How?

Bounding spheres and bounding rectangles

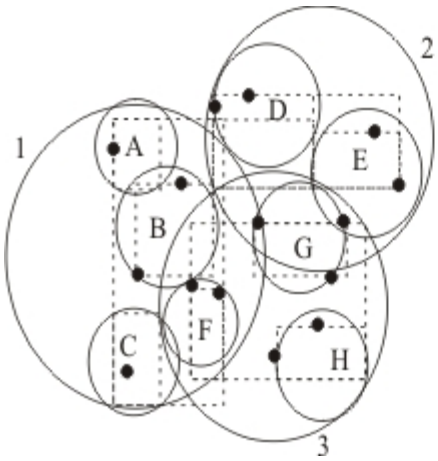
- SS-tree has regions with shorter diameters
- R*-tree has regions with smaller volumes
- How?
- Rectangles in d-dimensions
  - Diameter: Between 0 and $\sqrt{d}$
  - Volume: Between 0 and 1
- Spheres in d-dimensions
  - Diameter: Between 0 and $\sqrt{d}$
  - Volume: Between 0 and $(\sqrt{d}/2)^d(\pi^{d/2}/\Gamma(d/2+1)) \gg 1$

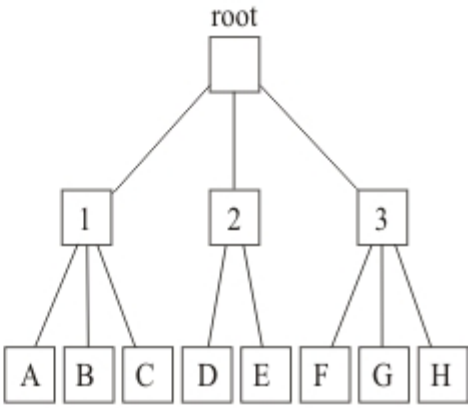
Bounding spheres and bounding rectangles

- SS-tree has regions with shorter diameters
- R*-tree has regions with smaller volumes
- How?
- Rectangles in d-dimensions
  - Diameter: Between 0 and $\sqrt{d}$
  - Volume: Between 0 and 1
- Spheres in d-dimensions
  - Diameter: Between 0 and $\sqrt{d}$
  - Volume: Between 0 and $(\sqrt{d}/2)^d(\pi^{d/2}/\Gamma(d/2+1)) \gg 1$
- Ideally, an index node should combine short diameter regions of SS-tree with small volume regions of R*-tree

SR-tree

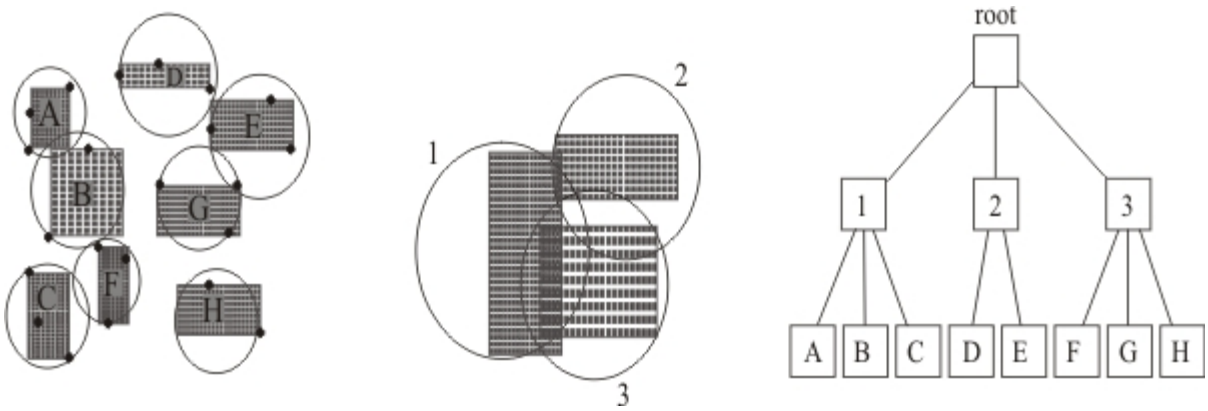
- Uses both bounding spheres and bounding rectangles
- Index node is intersection of these two
  - Tighter index
  - Implicitly stored





SR-tree

- Uses both bounding spheres and bounding rectangles
- Index node is intersection of these two
  - Tighter index
  - Implicitly stored



Structure and algorithms

- Index node explicitly maintains both
  - Bounding sphere
  - Bounding rectangle
- Center is weighted centroid of child nodes
- Radius is minimum of maximum distances to
  - Bounding spheres of child nodes
  - Bounding rectangles of child nodes
- Radius is tighter than SS-tree

Structure and algorithms

- Index node explicitly maintains both
  - Bounding sphere
  - Bounding rectangle
- Center is weighted centroid of child nodes
- Radius is minimum of maximum distances to
  - Bounding spheres of child nodes
  - Bounding rectangles of child nodes
- Radius is tighter than SS-tree
- When searching, minimum distance of query point to a node is maximum of minimum distances to
  - Bounding sphere of the node
  - Bounding rectangle of the node
- Similarly, maximum distance is minimum of maximum's
  - Better lower and upper bounds

Performance

- Storage cost is higher
    - 3 times that of SS-tree, 1.5 times that of R*-tree
- Fanout is lower
- Height is larger
- Number of internal node accesses is more
- Number of leaf node accesses is less
    - Better pruning of index nodes
- Together, faster search than SS-trees and R*-trees
- Construction time is larger

P-tree

- Index is an arbitrary bounding polyhedron defined by several hyperplanes
- Hyperplanes are normal to only certain orientation vectors
  - Basis vectors along with diagonals is a good choice
- Map convex polyhedron at d-dimensional attribute space to hyperrectangle at m-dimensional orientation space
- Polyhedra intersect if and only if the hyperrectangles intersect
- Searching involves checking the min and max of each axis in orientation space for intersection
- Orientation vectors may be added and deleted with no change to the index structure

Search performance and overlap

- Amount of overlap is directly related to search performance
- Definitions of overlaps

Overlap $$O = \frac{vol(\cup_{i,j,i \neq j}(R_i \cap R_j))}{vol(\cup_i R_i)}$$

Weighted Overlap $$WO = \frac{|\cup_{i,j,i \neq j} (R_i \cap R_j)|}{|\cup_i R_i|}$$

- Ideally, distribution of query should be known
  - When query is uniformly distributed, overlap measures multiple search path accesses
  - When query is estimated from data, weighted overlap is a better estimator

Search performance and overlap

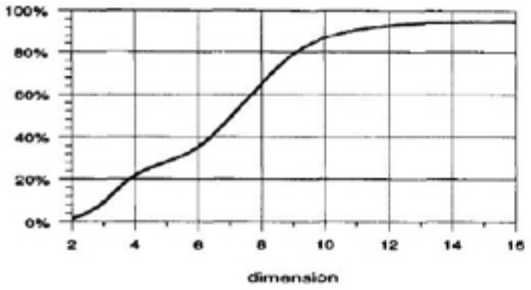
- Amount of overlap is directly related to search performance
- Definitions of overlaps

Overlap $$O = \frac{vol(\cup_{i,j,i \neq j}(R_i \cap R_j))}{vol(\cup_i R_i)}$$

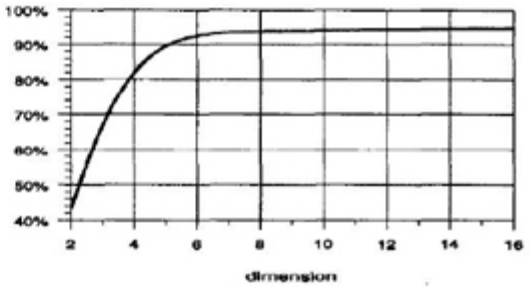
Weighted Overlap $$WO = \frac{|\cup_{i,j,i \neq j} (R_i \cap R_j)|}{|\cup_i R_i|}$$

- Ideally, distribution of query should be known
  - When query is uniformly distributed, overlap measures multiple search path accesses
  - When query is estimated from data, weighted overlap is a better estimator
- Multiple overlap: Counts the number of regions overlapping a certain volume
  - Should be an even better estimate

Effect of dimensionality on overlap



Overlap (uniform data)



Weighted overlap (real data)

- Data from R*-tree experiments
- Even for medium dimensionality, overlap approaches 100%
- As a result, search performance degrades rapidly