Module 5: Disk-based Index Structures

Lecture 20: R-trees

The Lecture Contains:

- Splitting a node
  - Exhaustive algorithm
  - Quadratic algorithm
  - Linear algorithm
  - Experiments on R-trees

Splitting a node

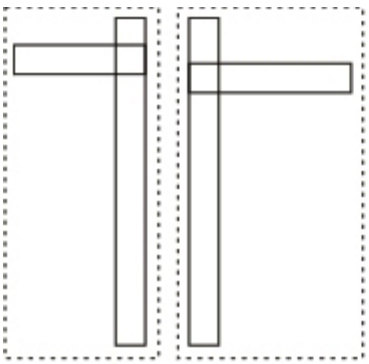
- Split a node with $m+1$ entries into two nodes having at least m entries each

Splitting a node

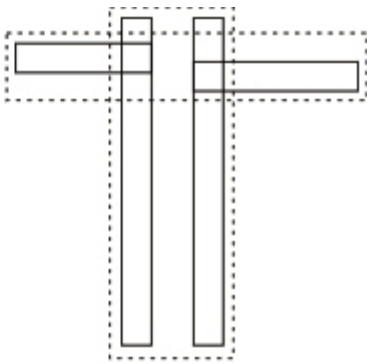
- Split a node with $m+1$ entries into two nodes having at least m entries each
- Minimize the probability that both the new nodes are accessed during a search
- Minimize the sum of probabilities of accessing the two nodes
    - Requires knowledge of query
- Probability is directly correlated to volume of MBR
    - Therefore, minimize *sum* of volumes

Splitting a node

- Split a node with $m+1$ entries into two nodes having at least m entries each
- Minimize the probability that both the new nodes are accessed during a search
- Minimize the sum of probabilities of accessing the two nodes
    - Requires knowledge of query
- Probability is directly correlated to volume of MBR
    - Therefore, minimize *sum* of volumes
- Three different strategies:
    - Exhaustive
    - Quadratic
    - Linear



Bad split



Good split

Exhaustive algorithm

- Examine all possibilities
    - Number of possibilities is exponential in m (and M)
- Choose the best
- Optimal
- Too slow and impractical

Quadratic algorithm

- Choose a pair of entries (seeds) that is the most wasteful
  - Choose entries ℓ_1 and ℓ_2 such that w = volume(ℓ) + (volume(ℓ_1) + volume(ℓ_2)) where ℓ is the covering hyper-rectangle of ℓ_1 and ℓ_2 is the largest
- ℓ_1 and ℓ_2 will be in different nodes
- If one group has $M - m + 1$ entries, put the rest into the other group to avoid under flow
- Otherwise, choose next entry ℓ_i to assign
  - For each remaining entry, calculate w_1 and w_2 as the increase in volume of node 1 and node 2 respectively
  - Pick ℓ_i with largest $|w_1 - w_2|$
- For ℓ_i, pick the group with the minimum volume increase
  - If tie, choose group with smaller volume, then lesser number of entries

Linear algorithm

- Seeds are chosen differently
- For each dimension, choose entries with highest minimum and lowest maximum
- Normalize the difference by the total range along the dimension
- Choose the entries with the largest difference as the seeds
- Rest of the algorithm is same as quadratic

Experiments on R-trees

- Insertion
  - Split time increases with page size
  - Decreases with increasing m

Experiments on R-trees

- Insertion
  - Split time increases with page size
  - Decreases with increasing m
- Deletion
  - Becomes very high with large m due to lots of re-inserts

Experiments on R-trees

- Insertion
  - Split time increases with page size
  - Decreases with increasing m
- Deletion
  - Becomes very high with large m due to lots of re-inserts
- Searching
  - Relatively insensitive to splitting algorithms
  - Less random I/O cost but more CPU cost with larger page size
  - Slight effect of m: increases with m
  - Size of index decreases with increasing m

Experiments on R-trees

- Insertion
  - Split time increases with page size
  - Decreases with increasing m
- Deletion
  - Becomes very high with large m due to lots of re-inserts
- Searching
  - Relatively insensitive to splitting algorithms
  - Less random I/O cost but more CPU cost with larger page size
  - Slight effect of m: increases with m
  - Size of index decreases with increasing m
- R-trees quite suitable for relational databases