Module 4: Index Structures

Lecture 14: Quadtrees and k-d-trees

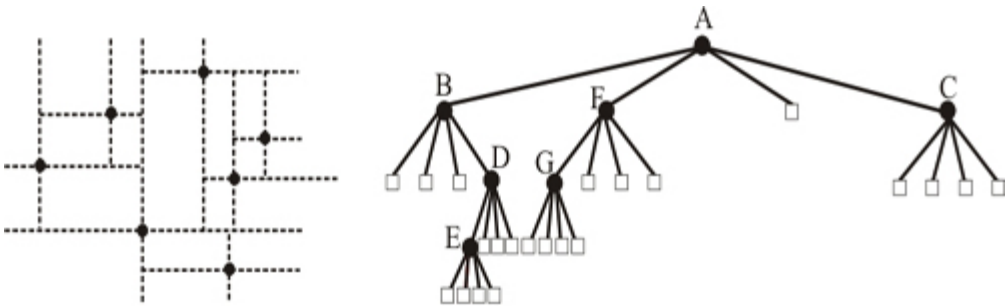
Prev topic | Next topic | Prev page | Next page

The Lecture Contains:

- Quadtree
  - Searching in quadtrees
  - Variants and extensions of quadtrees
- K-d-tree
  - Algorithms for k-d trees
  - Analysis of search time for k-d trees

Quadtree

- Two-dimensional binary search tree.
- Two-dimensional binary search tree.
- Each node contains a point from a two-dimensional space.
- Four children corresponding to 4 quadrants: NE, NW, SW, SE
- Unbalanced structure
- Insertion is simple
- Deletion is quite complicated
    - Simple re-insertion of all points is costly
- These are point quadtrees



Searching in quadtrees

- Point search requires angle determination
- Range search requires
    - Minimum bounding rectangle computation
    - Overlapped rectangles computation
- Time can be $O(2\sqrt{N})$ in worst case

Variants and extensions of quadtrees

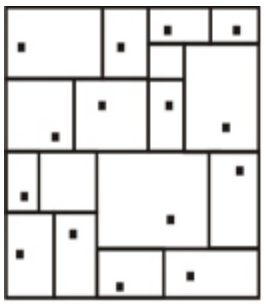
- Region quadtrees: Always divide at the geographical centre
- Can be extended to d dimensions
- Worst case searching time is $O(d.N^{(1-1/d)})$
- Partial range query runs in $O(s.N^{(1-1/d)})$ where s dimensions are specified.

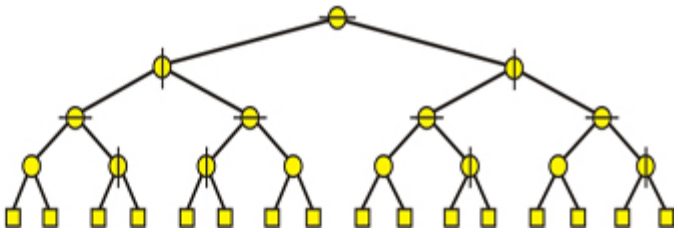
Variants and extensions of quadtrees

- Region quadtrees: Always divide at the geographical centre
- Can be extended to d dimensions
- Worst case searching time is $O(d.N^{(1-1/d)})$
- Partial range query runs in $O(s.N^{(1-1/d)})$ where s dimensions are specified.
- Called octrees in three dimensions.

K-d-tree

- Multidimensional unbalanced binary search tree
- Dimensions are ordered and splits are cycled through dimensions
- Split in level i is decided by "discriminator" based on dimension i mod k only
- Avoids 2^d fanout of quadtrees





Algorithms for k-d trees

- Bulk-loading takes $O(N \lg N)$ time
- Insertion takes $O(\lg N)$ time
- Deletion involves nding "suitable" i dimensional value from subtree
  - Since subtrees are ordered on other dimensions, this reduces to an exhaustive search.
- $O(\lg N)$ search time for point queries
- $O(d.N^{(1-1/d)} + T)$ search time for range queries where T is size of the answer set
- Partial range searching time for s dimensions is $O(s.N^{(1-1/d)})$

Analysis of search time for k-d trees

- Range can be thought of as two vertical lines and two horizontal lines
- Consider one vertical line
  - At root (assuming there is a vertical split), either the left subtree or the right subtree is traversed
  - The next level children have splits on y-dimension
  - Hence, both subtrees may be traversed
  - This yields the recurrence

$$Q(n) = 2 + 2Q(n/4)$$

Analysis of search time for k-d trees

- Range can be thought of as two vertical lines and two horizontal lines
- Consider one vertical line
    - At root (assuming there is a vertical split), either the left subtree or the right subtree is traversed
    - The next level children have splits on y-dimension
    - Hence, both subtrees may be traversed
    - This yields the recurrence

      $$Q(n) = 2 + 2Q(n/4)$$

    - Solving the above recurrence (with $Q(1) = O(1)$),

      $$Q(n) = O(\sqrt{N})$$

- Same time for horizontal lines
- Time for T points in the answer set is linear
- Worst-case searching time in 2 dimensions is $O(2\sqrt{N} + T)$

Module 4: Index Structures

Lecture 14:

Analysis of search time for k-d trees

- Range can be thought of as two vertical lines and two horizontal lines
- Consider one vertical line
    - At root (assuming there is a vertical split), either the left subtree or the right subtree is traversed
    - The next level children have splits on y-dimension
    - Hence, both subtrees may be traversed
    - This yields the recurrence

      $$Q(n) = 2 + 2Q(n/4)$$

    - Solving the above recurrence (with $Q(1) = O(1)$),

      $$Q(n) = O(\sqrt{N})$$

- Same time for horizontal lines
- Time for T points in the answer set is linear
- Worst-case searching time in 2 dimensions is $O(2\sqrt{N} + T)$

Previous Next