Module 7 - Dictionaries for Multi-Dimensional Data

Graham Cooper

July 2nd, 2015

Ranged Dictionaries

- insert/delete
- range search (look for everything within a range)

Skip Lists

Lists which contain a list on the bottom level of all of the nodes, and have a chance of adding 1 to the height of 1/2 until it fails. We can then check only the lists on each level and snake our way to the correct node looking for boundaries

Analysis of Skip Lists

If there are n nodes in S_i then it is expected to have $\frac{n}{2}$ at level S_{i+1} number of nodes in a skip list of n items is expected to be:

$$n + \frac{n}{2} + \frac{n}{4} + ... + 1 = n(1 + \underset{<2}{\frac{1}{2}} + ...) \in \Theta(n)$$

0.1 Search

Number of nodes visited of a list S_i = distance between the two consectuive nodes in $S_{i+1} \in \Theta(n)$ (1 + 1/2 + ...)
Number of levels $\in \Theta(logn)$ is what it is expected to be

Search $\in \Theta(logn)$

1 Range Search

A=

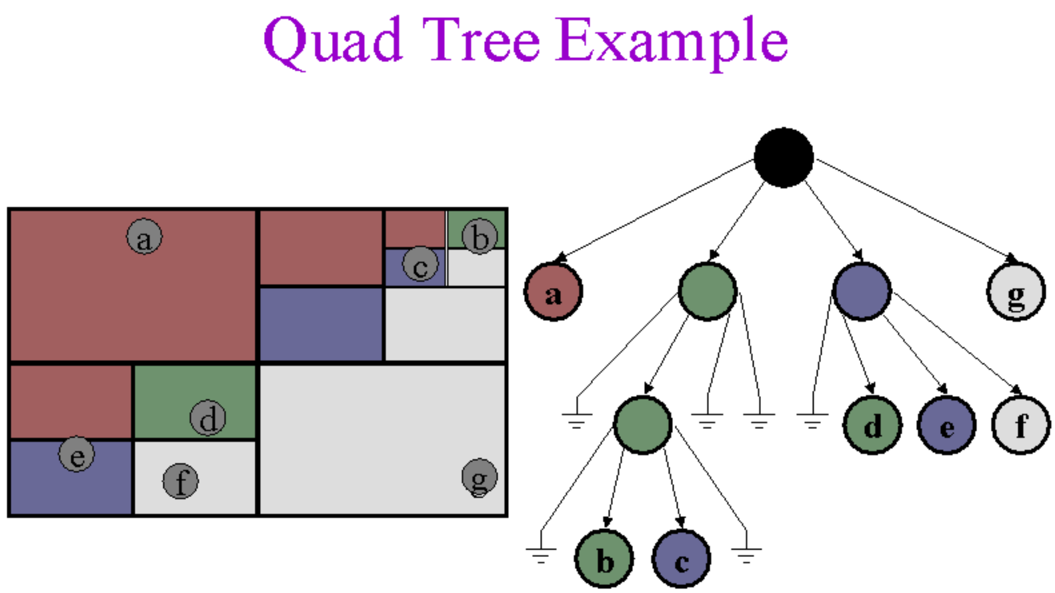
6	2	3	9	1	8	20	0

- Array(linked list) - RangeSearch takes $\Theta(n)$
- Sorted Array - BS the boundaries - $\Theta(logn + k)$ where k is the number of reported items. However, insert and delete takes $\Theta(n)$
- Balanced BST
  - Internal Nodes (check)
  - External Nodes X
  - Boundary Nodes ?

2Dimensional Range-Search

1. flatten two dimensions (like hashing) (rabge-search becomes hard)
2. Maintain a dictionary for each dimension (not effective)
3. Quad trees, kd-trees, Range-trees

Quad-Trees



Spread factor $= \frac{d_{max}}{d_{min}}$
height of quad tree $\Theta(log(\frac{d_{max}}{d_{min}})$

Why are quad-trees not good?
- We could have two nodes very close to each other, so we have to make many levels for very few nodes
3 points $\rightarrow$ h + 2 nodes in tree
4 points $\rightarrow$ 2h + 2 nodes in the tree
6 points $\rightarrow$ 3h + 2 nodes in the tree

kd-trees (cute trees, binary-trees)

We first split the keys in the area vertically, then draw lines out to the left and the right of the new root node, then we split each new sub-region horizontally, then vertically etc building our left/right nodes using the left/right sections and To/bottoms sections

Complexity of range search in KD-trees

= the number of visited nodes in the tree
= k + number of unsuccessful region checks $\leq$ k + number of regions that any vertical or horizontal line intersects * 4
= k + 4Q(n)

Q(n) = 2Q($\frac{n}{2}$) + 1

In each quadrant there are $\frac{n}{4}$ parts
Q(n) = 4Q($\frac{n}{16}$)+2
= 8Q($\frac{n}{64}$) + 3
Q(n) = $2^i Q(\frac{n}{2^{2i}}) + i$

Q(n) = $2^{\frac{1}{2} logn} + \frac{logn}{2}$
= $\sqrt{n} + \frac{logn}{2}$
= $O(\sqrt{n})$