lecture19: Fun with Trees

Sean Massung

Largely based on slides by Cinda Heeren
CS 225 UIUC

11th July, 2013
Announcements

- mp5.1 extra credit due tomorrow
- lab_huffman due Saturday (7/13)
- mp5 due Monday (7/15)
// when the function exits, the Vector parameter holds the sorted keys
// hint: use insertBack to add elements to the Vector
void sort(const TreeNode* subtree, Vector<K> & sorted)
{

}

// what is the running time?
// return true if this is an AVL tree. Assume you have a height
// function and that you know the tree is a BST
bool AVLCheck(const TreeNode* subtree)
{

// what is the running time?
// given a query key, return the key in the tree that is closest
K nearestKey(const TreeNode* subtree, const K & query, const K & best) {

}  

// what is the running time?
rangeQuery

// prints all keys on the interval [a, b]
void rangeQuery(const TreeNode* subtree, const K & a, const K & b) {

}

// what is the running time?
// return the lowest node in the tree that could be a
// parent to both a and b
K LCA(const TreeNode* subtree, const K & a, const K & b) {

}  

// what is the running time?
Red-Black Trees

RB trees are less rigidly balanced than AVL trees; this leads to faster insertion times, but slower retrieval.

- Find: $O(\log n)$, Insert: $O(\log n)$, Delete: $O(\log n)$
- Rotations are performed to keep the RB properties enforced:
  1. A node is either red or black
  2. The root is black
  3. All leaves are black
  4. Both children of every red node are black
  5. Every simple path from a node to its descendant leaves contains the same number of black nodes
Splay Trees

- Find: $O(\log n)$*, Insert: $O(\log n)$*, Delete: $O(\log n)$*
- Whenever find or insert is called, rotate that key to the root (how?)
- What kind of data will benefit from such a structure?
Scapegoat Trees

- Find: $O(\log n)$, Insert: $O(\log n)\ast$, Delete: $O(\log n)\ast$
- Ensures that $\forall T,$
  - $\text{size}(T_{\text{left}}) \leq \alpha \cdot \text{size}(T)$,
  - $\text{size}(T_{\text{right}}) \leq \alpha \cdot \text{size}(T)$,
  - $\alpha \in [.5, 1]$
- If this is not satisfied, the entire subtree is rebuilt to be completely balanced in $O(n)$ time
B-Trees

- *Out-of-core* data structure
- $O(\log n)$ running times, but also allows sequential access in nodes
- Designed mainly for use in filesystems or databases (more in CS 411)
- The tree structure is built to minimize the number of expensive disk seeks
- Built by splitting large nodes in half, creating a new parent
Quadtree and KD-trees

mp5 and mp6!