CSE 530A

Binary Search Trees

Washington University
Fall 2013
Indexing

• A common index type is based on the B-tree
  – The B-tree data structure is a generalization of the Binary Search Tree
    • A binary search tree is an ordered binary tree
Binary Trees

- A binary tree is a tree in which each node has at most two child nodes.
- The (single) node without a parent is the root.
- The nodes without children are the leaves.
- The depth of a node is the length of the path from the root to the node.
- The height of a tree is length of the longest path.
- Siblings are nodes that have the same parent.
- A node x is an ancestor of a node y if x exists on the path from the root to y.
  - And y is then a descendant of x.
- The in-degree of a node is the number of incoming edges.
- The out-degree of a node is the number of outgoing edges.
Binary Search Trees

• A binary search tree is an ordered binary tree

• That is, it is a binary tree where
  – The left subtree of a node with key x contains only nodes with keys < x.
  – The right subtree of a node with key x contains only nodes with keys > x.
  – Both the left and right subtrees are binary search trees.
Binary Search Trees

• Searching for node $x$
  – Start at root
  – Recursively descend the tree going left if $x <$ current node or right if $x >$ current node
  – Eventually will either find a node $x$ or end up at a node with no child on the needed side
Binary Search Trees

• Inserting node $x$
  – (Node $x$ must not already exist)
  – Start at root
  – Recursively descend the tree going left if $x < \text{current node}$ or right if $x > \text{current node}$
  – Eventually will end up at a node with no child on the needed side
  – Add new node as leaf
Binary Search Trees

• Deleting node x
  – Three cases
    • Node x is a leaf
      – Delete the leaf node
    • Node x has one child
      – Remove node x and replace it with its child
    • Node x has two children
      – Choose either x's in-order predecessor or successor (call it y)
      – Replace x with y and delete the old y
Binary Search Trees

- Replace 4 with 5 and delete 5
Binary Search Trees

• In the average case, search, insertion, and deletion are all $O(\log n)$
• In worst case they are $O(n)$
  – Can make the worst case also $O(\log n)$ by using a balanced binary tree
Balanced Binary Trees

- A balanced binary tree is a tree where the height of the two subtrees of any node differ by at most one
  - This requirement is usually relaxed to be where no leaf is "much farther" away from the root than any other leaf
  - Definition of "much farther" varies by balancing scheme
Red-Black Trees

• A red-black tree is a balanced binary tree where
  1. All nodes are colored red or black.
  2. The root is black.
  3. All NIL leaves are black.
  4. Both children of every red node are black.
  5. The path from a node to any descendent leaf contains the same number of black nodes as to any other descendent leaf.
Red-Black Trees

• It is convenient to think of all leaving being special empty (NIL) leaves
  – This means all nodes have either two children or zero children
Red-Black Trees

• These properties mean that the path from the root to the farthest leaf is no more than twice that to any other leaf

• Search, insertion, and deletion are all $O(\log n)$
  – Note that the space requirement, like most binary search trees, is $O(n)$
Red-Black Tree Insertion

• Insertion
  – Start by inserting into the tree as with a general binary search tree
  – "Fix" the tree if any of the red-black properties are violated
Red-Black Tree Insertion

• Case 1: The newly inserted node is the new root
  – Fix tree by coloring the new node black
• Case 2: The new node's parent is black
  – Tree is OK, no fix needed
Red-Black Tree Insertion

• Case 3: Both the parent and the uncle of the new node are red
  – Color the new node red
  – Change the parent and uncle to black
  – Change the grandparent to red
  – Recursively fix the tree now treating the grandparent as the new node
Red-Black Tree Insertion

• Case 4:
  – The parent of the new node is red but the uncle is black and
  – The new node is the right child of its parent and
  – The parent node is the left child of its parent
    • Perform a left rotation on the parent
    • Recursively fix the tree treating the old parent as the new node

Treat the symmetric case (where N is the left child of P and P is the right child of G) by doing a right rotation.
Red-Black Tree Insertion

• Case 5:
  – The parent is red but the uncle is black and
  – The new node is the left child of its parent and
  – The parent node is the left child of its parent
    • Perform a right rotation on the grandparent
    • Switch the colors of the parent and grandparent

Treat the symmetric case (where N is the right child of P and P is the right child of G) by doing a left rotation.