

# For Wednesday

- Read Weiss, chapter 4, sections 7-8
- Homework:
  - Chapter 4, exercise 19

# Paper 1

- Any questions?

# Quicksort Homework

# Problem With Binary Trees

- Basic operations are  $O(h)$
- What is the height?
- Best case --
- Worst case --

# Solution

- How we can solve the performance problems of binary search trees?

# AVL Search Trees

- An AVL tree is a binary tree such that:
  - the difference between the heights of the left subtree and the right subtree is no more than one
  - the left subtree and the right subtree are AVL trees
- An AVL search tree is simply a binary search tree which is also an AVL tree

# Maintaining the AVL Properties

- Each node has a balance factor associated with it -- 0, 1, or -1
- If the nodes subtrees have equal heights, the balance factor is 0
- If the right subtree has greater height (by 1) the balance factor is -1
- If the left subtree has greater height (by 1) the balance factor is 1

# Binary Search Tree Operations

- Search
  - Time Complexity
- Insertion
  - May have to rotate
  - Time Complexity
- Deletion
  - May have to rotate up to  $\log(n)$  times
  - Time Complexity

# Splay Trees

- Interested in the cost of a sequence of search operations rather than the cost of a single search.
- We want to make sure that the **amortized** cost of  $M$  search operations is  $M \log N$ .

# Basic Idea

- When we find a node, we're going to rotate it to the top in a way that helps to balance the tree if it is currently unbalanced.

# Cases

- Found node has no grandparent: rotate node and root
- Found node has a grandparent:
  - zig-zig case (parent is same side of grandparent that node is of root): rotate node and grandparent
  - zig-zag case (node's value is in-between value of parent and grandparent): do a standard AVL double rotation

# Comparison

- Splay trees and AVL trees