

# For Wednesday

- Read Weiss, chapter 4, section 5
- Homework:
  - Chapter 4, exercise 9

# Programming Assignment 2

- Any questions?



# Binary Search Trees

- An addition to our possible dictionary implementations
- We're interested in operations:
  - Search
  - Insert
  - Delete
  - Output in ascending order of keys
  - FindMin
  - FindMax

# Definition

- Binary tree
- Each element has an associated key
- Keys are not duplicated
- For any node, its left child (if any) has a key smaller than its key and its right child (if any) has a key larger than its key

# Binary Search Tree Operations

- Search
  - Time Complexity
- Insert
  - Time Complexity
- Deletion
  - Of a leaf
  - Of a node with one non-empty subtree
  - Of a node with two non-empty subtrees
  - Time Complexity
- Output Ordered List
- FindMin
- Find Max

# 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