Administrivia
• HW#3 due today
  – If people don’t mind, I might rearrange this to 25 points…
• HW#4 out
  – Start earlier! Don’t make last-minute appointments – it makes my life hard

Agenda
• Continue trees

Binary search trees
• What’s a binary tree?
  – Two children, always
• Main concept:
  – Max(left subtree) must be < current node, min(right subtree) must be > current node
• Why?
  – Combines advantages of a linked list and an ordered array
  – Can insert fast and search fast
  – Unlimited growth
  – Relatively fast indexed access

Writing the Tree in Java
• “Node” class, with left and right children
• Data in node as well
• Very similar to Link
• Main “Tree” class that links to root, with find, insert, delete, etc. methods

Operations in a BST
• Search
  – Simple: walk left or right depending if < or > than current
  – If we hit the bottom, we can’t find it
  – O(log N) time
• Insert
  – “Search”, and then put in the appropriate place
  – Need a “current” and a “parent” pointer, similar to linked-list

Traversing the tree
• Unlike search, want to walk in an abstract order, sort of like arrays
• Three means of traversal; all recursive
  – Inorder
    • Visit left subtree
    • Visit node
    • Visit right subtree
  – Preorder
- Postorder
- The latter two have use in expressions (pg. 386)

### 8 Other operations
- Min/max values
- Deleting a node
  - More complicated!
  - If no children, then nuke
  - One child
  - More than one child
    - Make one left, and go all the way right, or;
    - Make one right, and go all the way left
    - Take that node and put it at the deleted node’s location
      - Move the right child of the moved node up one notch
    - Book uses latter convention

### 9 Tree complexity
- # of levels of a full tree is log N
  - Search, insert, delete is O(log N)
- What if it isn’t full? Difficult analysis
  - Insert(1)
  - Insert(2)
  - ...
  - In fact, this is the one downside of simple BST trees: easy to make unbalanced
  - There are alternatives; you can read chapter 9 should you like

### 10 Next time
- Finish Trees
- Begin Hashing