CSSE 230 Day 15

AVL insert/Delete Review
AVLTree practice
Worktime

After today, you should be able to…
…write code to insert an item into an EditorTree using rank and keep it balanced

After insertion into AVL or EditorTree, go up tree, updating balance codes and checking for imbalance

- p = parent of inserted node
- while p != null
  - // 3 cases (=, tipped towards, tipped away)
  - if p.balanceCode is '='
    - set code to '/' or '\' (towards insertion point)
    - p = p.getParent()
  - else if p.balanceCode indicates "insertion was in shorter subtree"
    - change code to '='
    - Break (STOP)
  - else //insertion was into taller side.
    - do the appropriate rotation
    - Break (STOP)

This is for Milestone 1; You will design a similar procedure for deletion (milestone 2)

Term Project: EditorTrees

Like BST, except:

1. Keep height-balanced
2. Insertion/deletion by **index**, not by comparing elements.
So not sorted

Examples:

▸ EditorTree et = new EditorTree()
▸ et.add('a') // append to end
▸ et.add('b') // same
▸ et.add('c') // same. Rebalance!
▸ et.add('d', 2) // where does it go?
▸ et.add('e')
▸ et.add('f', 3)

▸ Notice the tree is height-balanced (so height = O(log n)), but not a BST

What is the goal of EditorTrees?
Implementing the List ADT using a balanced tree.

- Get/Insert/delete by **index**
  - all in O(log n) time
  - .add(item) adds to end
  - .add(item, index) adds it to the given index, so the position of the item at that index (and all to the right) increases by 1

- Efficient size and height
  - using rank or maintaining fields

- Plus:
  - Concatenate/Split, like String + and .substring()

Today's agenda

▸ Discuss **rank** and do quiz on it.

▸ Make sure your whole team has finished and understands yesterday's AVL quiz
  ◦ Get them checked off

▸ Work with your team on the project
  ◦ I expect to see you working on paper (designing your algorithms and understanding tests) as much as (or more than) on the computer