Nick Palumbo
Homework 3
10/1/2015

Complete the following textbook exercises:

Chapter 8, section 6 (pg 352)
R-8.1
R-8.7
R-8.9
R-8.20
C-8.32

Chapter 11, section 7 (pg 528)
R-11.2
R-11.5
R-11.8
R-11.11
C-11.40

R-8.1
a.Which node is the root?

/user/rt/courses/

b. What are the internal nodes?

cs016/
homework/
programs/
cs252/
projects/
grades
papers/
demos/

c. How many descendants does node cs016/ have?

descendants = 9

d. How many ancestors does node cs016/ have?

ancestors = 1
which is the root node

e. What are the siblings of node homeworks/?

grades
programs/

f. Which nodes are in the subtree rooted at node projects/?

papers/
demos/
buylow
sellhigh
market

g. What is the depth of node papers/?

depth = 3

h. What is the height of the tree?

height = 4

R-8.7
What are the minimum and maximum number of internal and external nodes in an improper binary tree with n nodes?

A binary tree is proper if each node has either zero or two children.

The maximum number of internal nodes for an improper tree is (n - 1) internal nodes and 1 external node.

The minimum number of internal nodes for an improper tree is n/2 (n > 0)

R-8.9
Give a proof by induction of Proposition 8.9.

Proposition 8.9: In a nonempty proper binary tree T , with nE external nodes and nI internal nodes, we have nE = nI + 1.

show that for any particular n >= 1, there is a finite sequence of implications that starts with someone known to be true and ultimately leads to showing that q(n) is true.

1. show that q(n) is true for n = 1
2. inductive step:
        if q(j) is true for all j < n, then q(n) is true

F(n) = n + 1

nE > nI

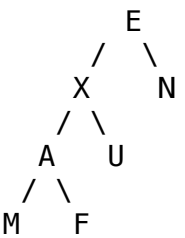
Base cases: (n = 0) F(0) = (0) + 1 = 1 ; 1 > 0
                     F(n) = n + 1   ; n + 1 > n

Induction step: (n > 0)
F(n) = number of external nodes with n internal nodes

x < n
n >= 1

F(n) = n + 1
F(n) = F(x) + F(n − x − 1)
      = x + 1 + n − x − 1 + 1
      = n + 1

R-8.20
Draw a binary tree T that simultaneously satisfies the following:
• Each internal node of T stores a single character.
• A preorder traversal of T yields EXAMFUN.
• An inorder traversal of T yields MAFXUEN.



C-8.32

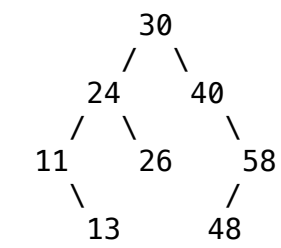
Let T be a (not necessarily proper) binary tree with n nodes, and let D be the sum of the depths of all the external nodes of T . Show that if T has the minimum number of external nodes possible, then D is O(n) and if T has the maximum number of external nodes possible, then D is O(n log n).

D is O(n) if T has the minimum number of nodes possible because the depths must all be checked on each individual external nodes making the runtime O(n).

D is O(n log n) if T has the maximum number of nodes possible because the binary tree only needs to have depth computed once for each height which adds the sum of the heights of all external nodes. Then the n is for checking how many external; nodes are at that depth/height.

R-11.2

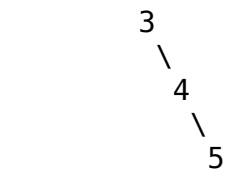
Insert, into an empty binary search tree, entries with keys 30, 40, 24, 58, 48, 26, 11, 13 (in this order). Draw the tree after each insertion.

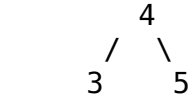


R-11.5

Dr. Amongus claims that the order in which a fixed set of entries is inserted into an AVL tree does not matter—the same AVL tree results every time. Give a small example that proves he is wrong.

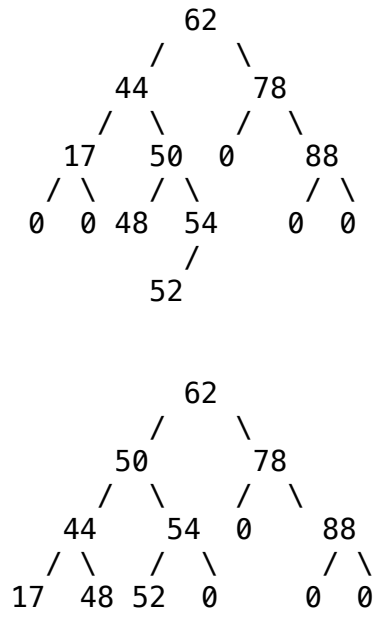
insertion of {3, 4, 5} will not be the same binary tree as insertion of {4, 3, 5}.





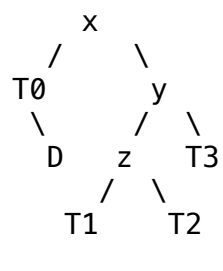
R-11.8

Draw the AVL tree resulting from the insertion of an entry with key 52 into the AVL tree of Figure 11.14b.



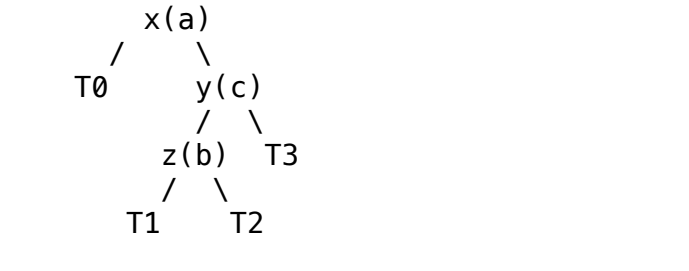
R-11.11

Give a schematic figure, in the style of Figure 11.13, showing the heights of subtrees during a deletion operation in an AVL tree that triggers a tri-node restructuring for the case in which the two children of the node de-noted as y start with equal heights. What is the net effect of the height of the rebalanced subtree due to the deletion operation?



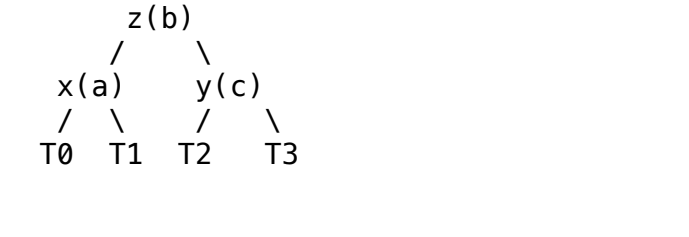
x : h = 4
T0: h = 2
y : h = 3
z : h = 2
T3: h = 1
D : h = 1
T1: h = 1
T2: h = 1

delete D



x : h = 4
T0: h = (1)
y : h = 3
z : h = 2
T3: h = 1
– : ————
T1: h = 1
T2: h = 1

perform tri-node reconstruction to rebalance AVL tree



x : h = (2)
T0: h = (1)
y : h = (2)
z : h = (3)
T3: h = (1)
T1: h = 1
T2: h = 1

The net effect of the height is that the total height is lowered.

C-11.40

In our AVL implementation, each node stores the height of its subtree, which is an arbitrarily large integer. The space usage for an AVL tree can be reduced by instead storing the balance factor of a node, which is defined as the height of its left subtree minus the height of its right subtree. Thus, the balance factor of a node is always equal to -1, 0, or 1, except during an insertion or removal, when it may become temporarily equal to -2 or +2. Reimplement the AVLTreeMap class storing balance factors rather than subtree heights.

```
if balance factor(root) = 2:
     if balance factor(right) = 1:
          self._rotate(left)
     else if balance factor(right) = -1:
          self._rotate(left)
           self._rotate(left)
else:
     if balance factor(left) = 1:
          self._rotate(left)
     else if balance factor(left) = -1:
          self._rotate(left)
           self._rotate(left)
```