Kyle Nealy
[redacted]
10/2/15
C343
HW3

R-8.1) a.) /user/rt/courses/

b.) [/user/rt/courses/], [cs016/], [cs252/],
[homeworks/], [programs/], [projects/],
[papers/], [demos/].

- any node whose key does not have "/" as a suffix.

c.) 9

d.) 1

e.) [grades] and [programs/]

f.) [papers/], [demos/],
[buylow], [sellhigh], [market].

g.) 3

h.) 4

R-8.7)

	Min	Max
Internal	[redacted] $n-n$	$n-1$
External	n^0	n

For improper trees, in the case of internal nodes, consider a tree with only a root node, thus $n-n = 1-1 = 0$ internal nodes for the minimum. Consider a tree with n right children, thus the max # of internal nodes is $n-1$.

In the case of external nodes, if the tree has only a root, n^0 [redacted] and n external nodes = 1 node.

[redacted]

R-8.9

T n_e external n_i internal Induction

$n_e = n_i + 1$

- Base case: only a node root $n_e = 1$, $n_i = 0$

- Assume: true w/ K nodes $[n_e = n_i + 1]$

- at height h: $\left[n_e = 2^h,\ n_i = \sum_{x=0}^{h-1} 2^x \right] + 2^h$

- at height h+1 $\left[n_e = 2^{h+1},\ n_i = \sum_{x=0}^{h} 2^x \right]$

Formula holds at height h, thus it holds for h+1.

Base Case:

K nodes $n_e = n_i + 1$

- adding 2 more nodes

$n_e' = n_e + 2 - 1$

$n_i' = n_i + 1$

Inductive Assumption:

$n_e = n_i + 1$

$n_e' = n_e + 1$

$= n_i + 1 + 1$

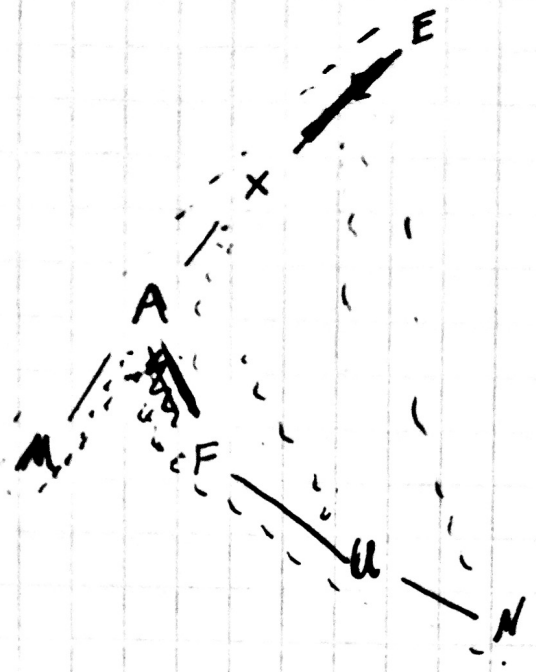
$= n_i' + 1$

Thus, formula holds for K nodes → K+1 nodes.

Therefore:

In a non-empty proper binary tree T, with n_e external nodes and n_i internal nodes, the external nodes n_e = internal nodes $n_i + 1$.

R-8.20



Preorder: EXAMFUN

Postorder: MAFXUEN

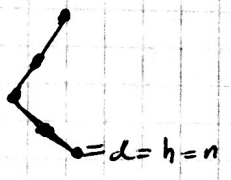
C- 8.32)

n nodes

D sum of depths

T min external $\quad D \in O(n)$

T max external $\quad D \in O(n \log n)$

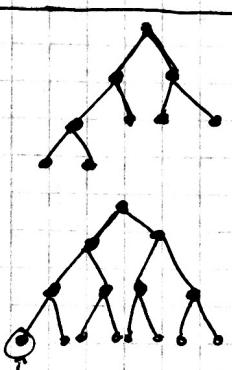


$= d = h = n$

$h, c \quad D \leq Cn$

$h_0 = 0 \quad C = 1$

$D \leq n$ ✓ $\quad D = n$



$d_c = h - 1 = \log n$

$h_c = 2^{\log n} = n$

$D \leq n \log n$

$D \leq n \log n$

Complete the tree by adding nodes to bottom level

n = nodes

h = height

$K = 1, C = 1$

$D \leq C \cdot n$

$D \leq n \log n \leq n,$

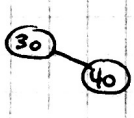
Thus, if T has minimum possible external nodes, then $D \in O(n)$, and if T has maximum external nodes possible, then $D \in O(n \log n)$.

R.-11.2

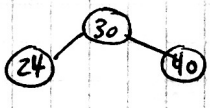
T = blank tree

- Insert(30): T =

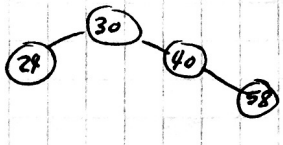
- Insert(40): T =



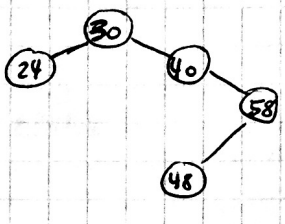
- Insert(24): T =



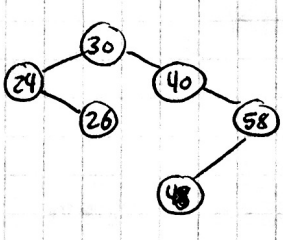
- Insert(58): T =



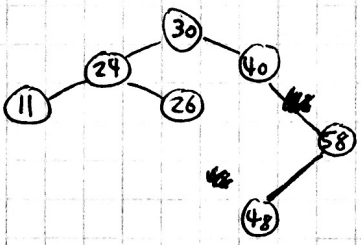
- Insert(48): T =



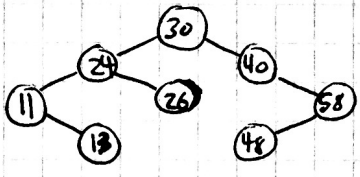
- Insert(26): T =



- Insert(11): T =



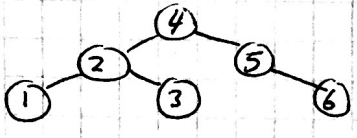
- Insert(13): T =



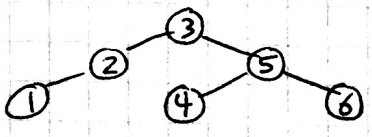
R.-11.5

Given the set of entries $\{1, 2, 3, 4, 5, 6\}$:

If inserted into an empty AVL tree in the order 1, 2, 3, 4, 5, 6, results in this tree:

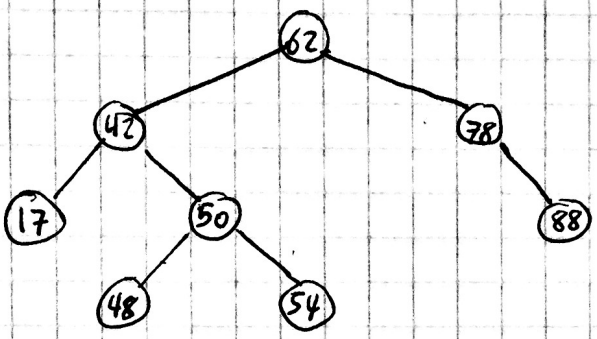


If inserted into an empty AVL tree in the order 6, 5, 4, 3, 2, 1 results in this tree:

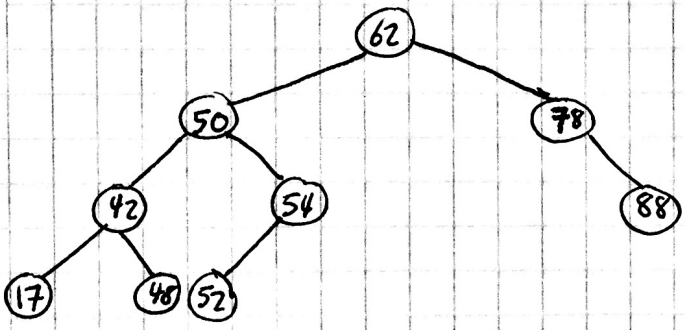


R-11.8

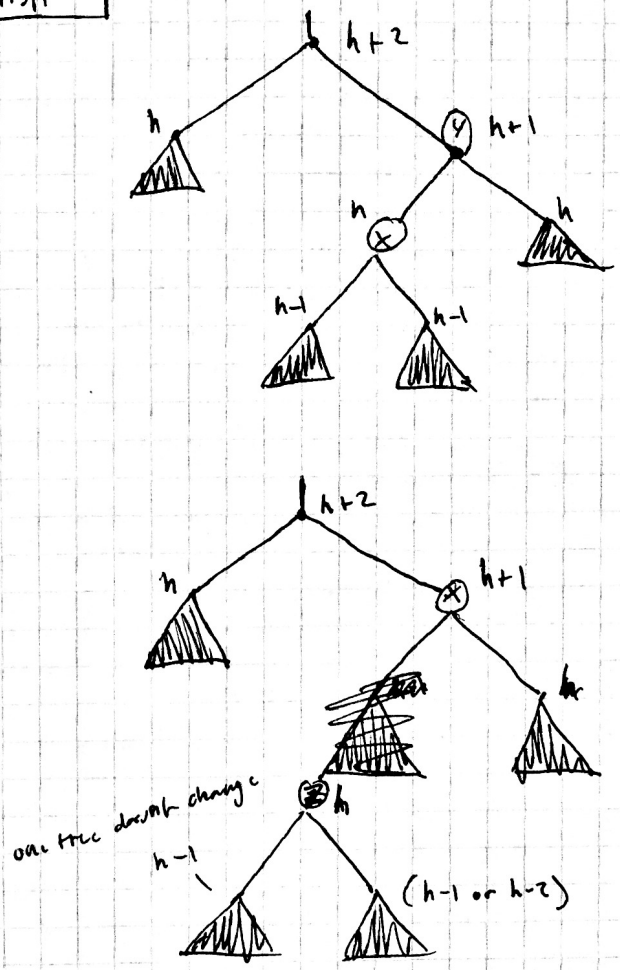
AVL Tree:



Insert node (52):



R-11.11



Net Effect:

Doesn't change the height of tree overall, unless there is only 1 subtree, or the tree only has 1 node.

C-11.40

```
class AVLTreeMap(TreeMap):

    class _Node(TreeMap._Node):

        __slots__ = '_balance'

        def __init__(self, element, parent=None, left=None, right=None):
            super().__init__(element, parent, left, right)
            self._balance = (self.left_height - self.right_height)

        def left_height(self):
            return self._left._height if self._left is not None else 0

        def right_height(self):
            return self._right._height if self._right is not None else 0

        ...
```

(All else stays ↓ the same in original code)