Note Title 2/13/2009

6.006 RECITATION 4 13 Feb 2009

* Binary Search Trees
* Lecture 4
* AVL Trees

AVL TREES

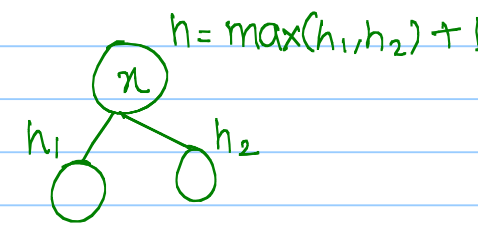
G.M. Adelson-Velsky and E.M. Landis
"An algorithm for the organization of information", 1962

Balanced Binary Search Tree

For any node x, height(x) : Length of longest path from x to a leaf node

height(x) = max(height(left(x)), height(right(x))) + 1

height(NULL) = -1, height(y) = 0 $\forall y$: leaf node



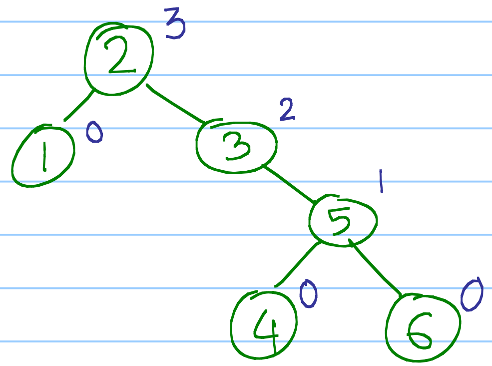
AVL invariant

$\forall x$: node, $|\text{height}(\text{left}(x)) - \text{height}(\text{right}(x))| \le 1$ $\quad |h_1 - h_2| \le 1$

$\text{height}(\text{root}) < 2\log n$

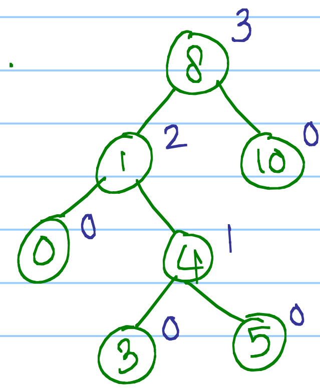
EXCERCISE

A.



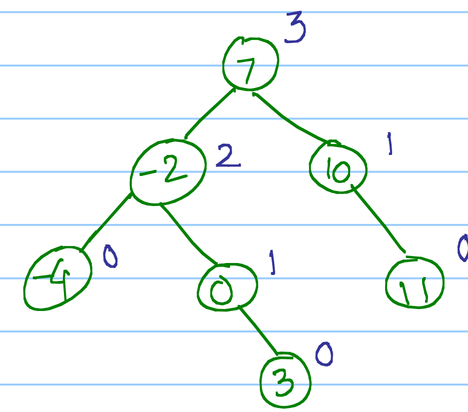
AVL ✗

B.



AVL ✗

C.



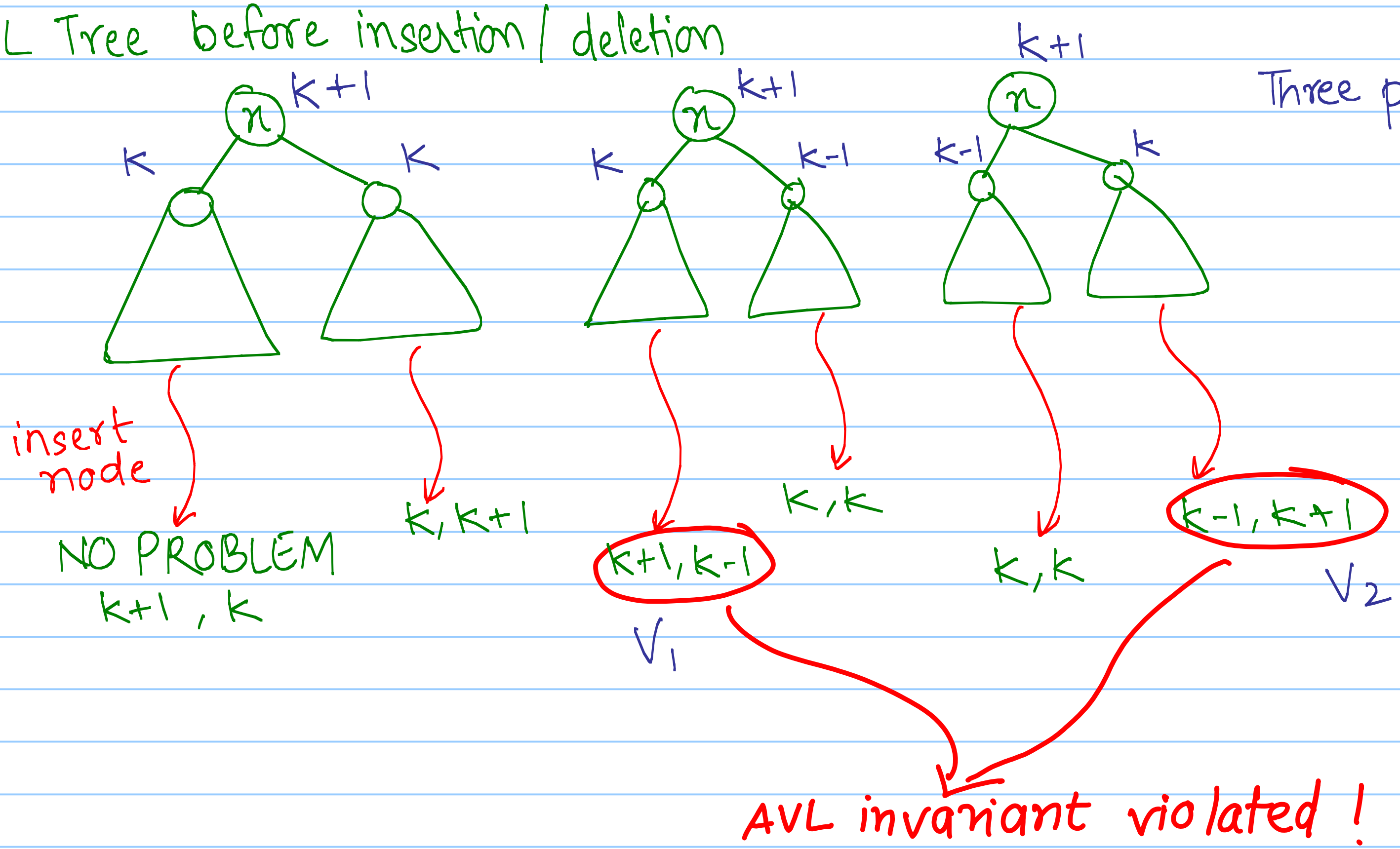
AVL ✓

$$N_h \geq N_{h-1} + N_{h-2} + 1$$
$$\geq 2N_{h-2} + 1$$
$$\geq 1 + 2(1 + 2N_{h-4})$$
$$\geq 1 + 2 + 2^2(1 + 2N_{h-6})$$
$$\geq 1 + 2 + 2^2 + \ldots\ldots 2^{h/2} = 2^{h/2+1} - 1$$
$$N_h \geq 2^{h/2}$$
$$h \leq 2 \log N_h$$

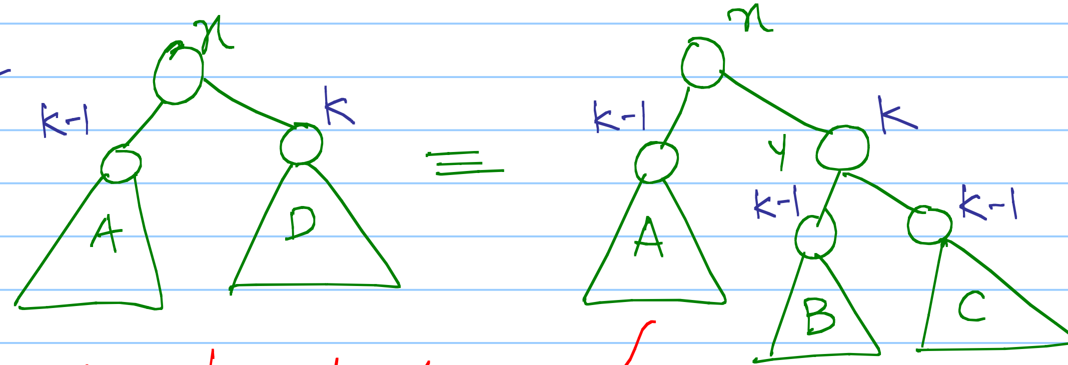
ROTATIONS

AVL Tree before insertion / deletion

Three possible cases

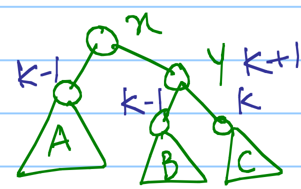


Consider a violation V_2

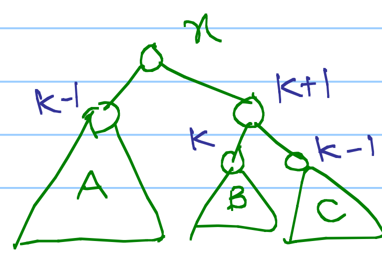


insert node α

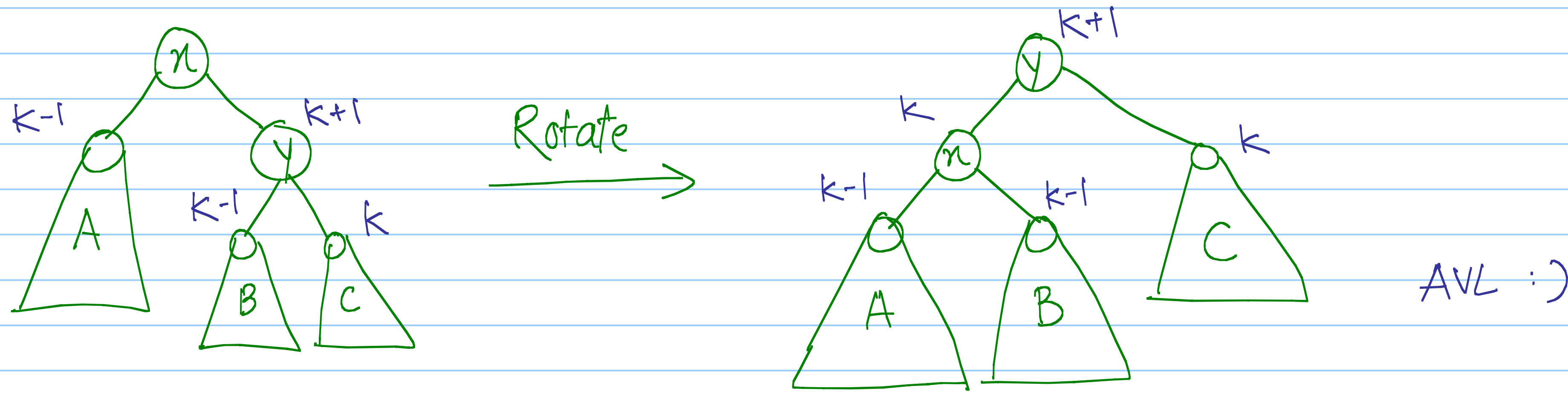
Case 1: goes to C



Case 2: goes to B

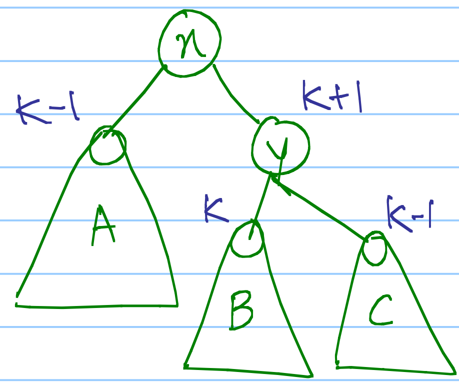


Case 1:

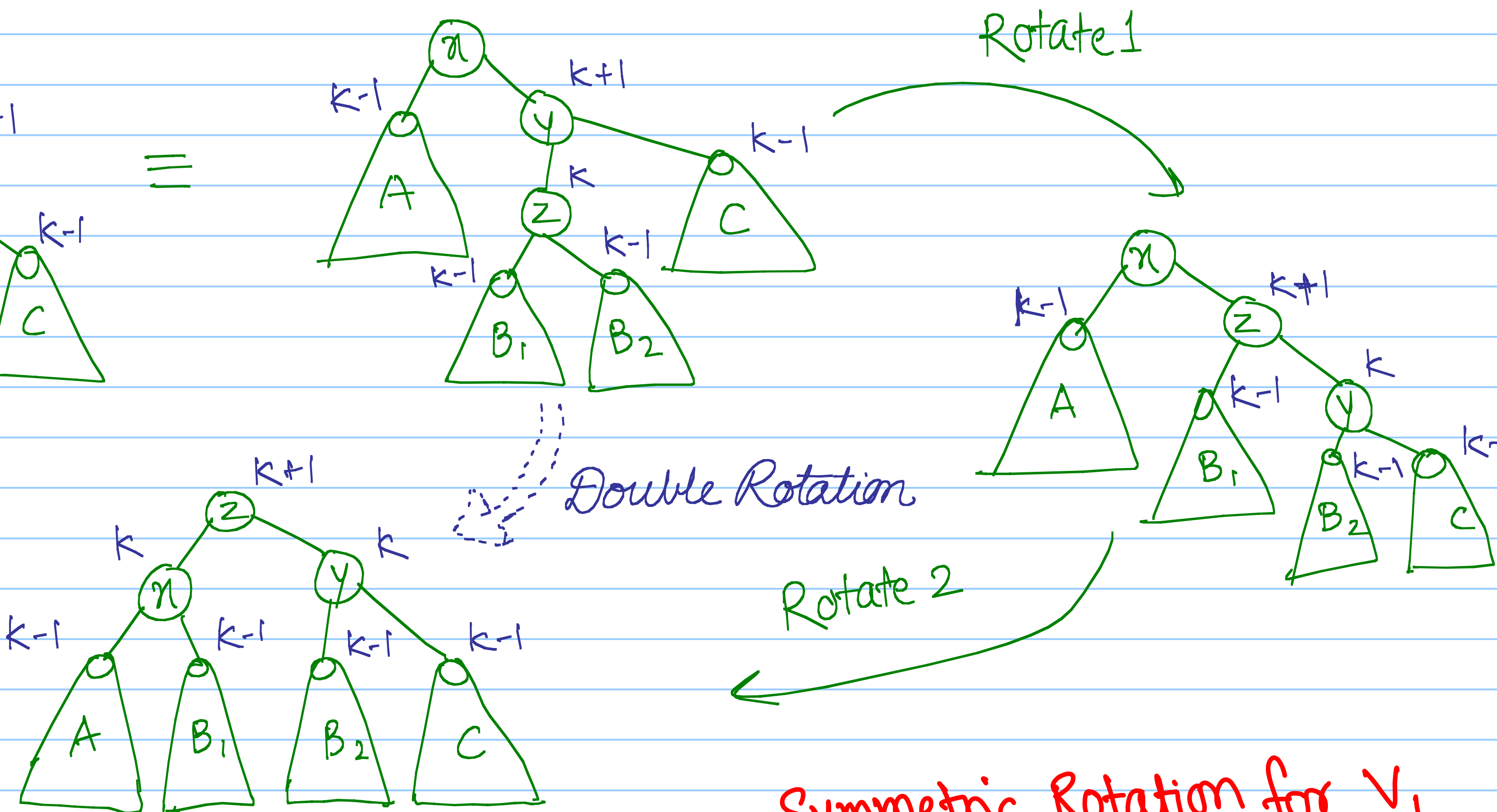



```
def left_rot(x):
    y = x.right
    x.right = y.left
    y.left = x
    return y
```

Case 2:

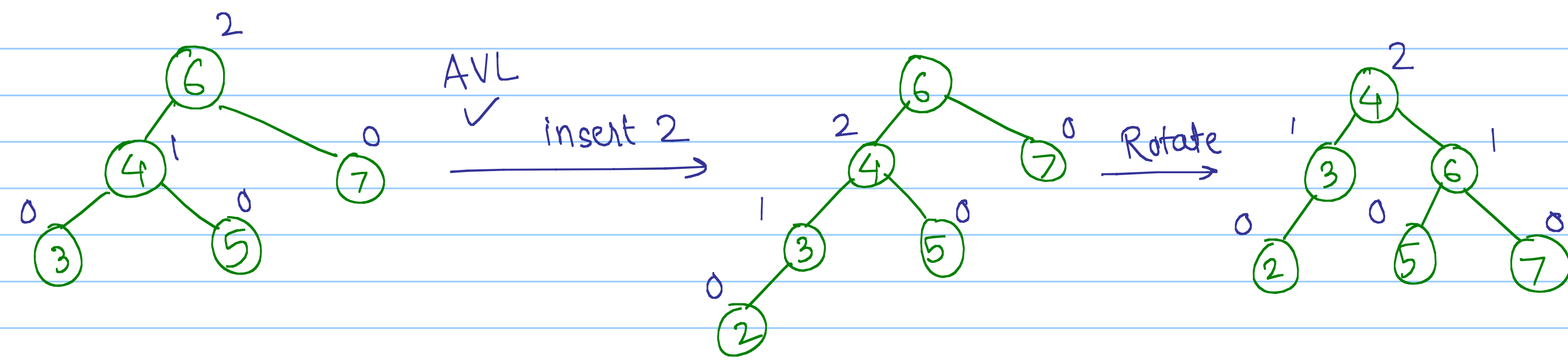


=




```
def dbl_left_rot(x):
    y = x.right
    z = y.left
    x.right = z.left
    z.left = x
    y.left = z.right
    z.right = y
    return z
```

Symmetric Rotation for V_1



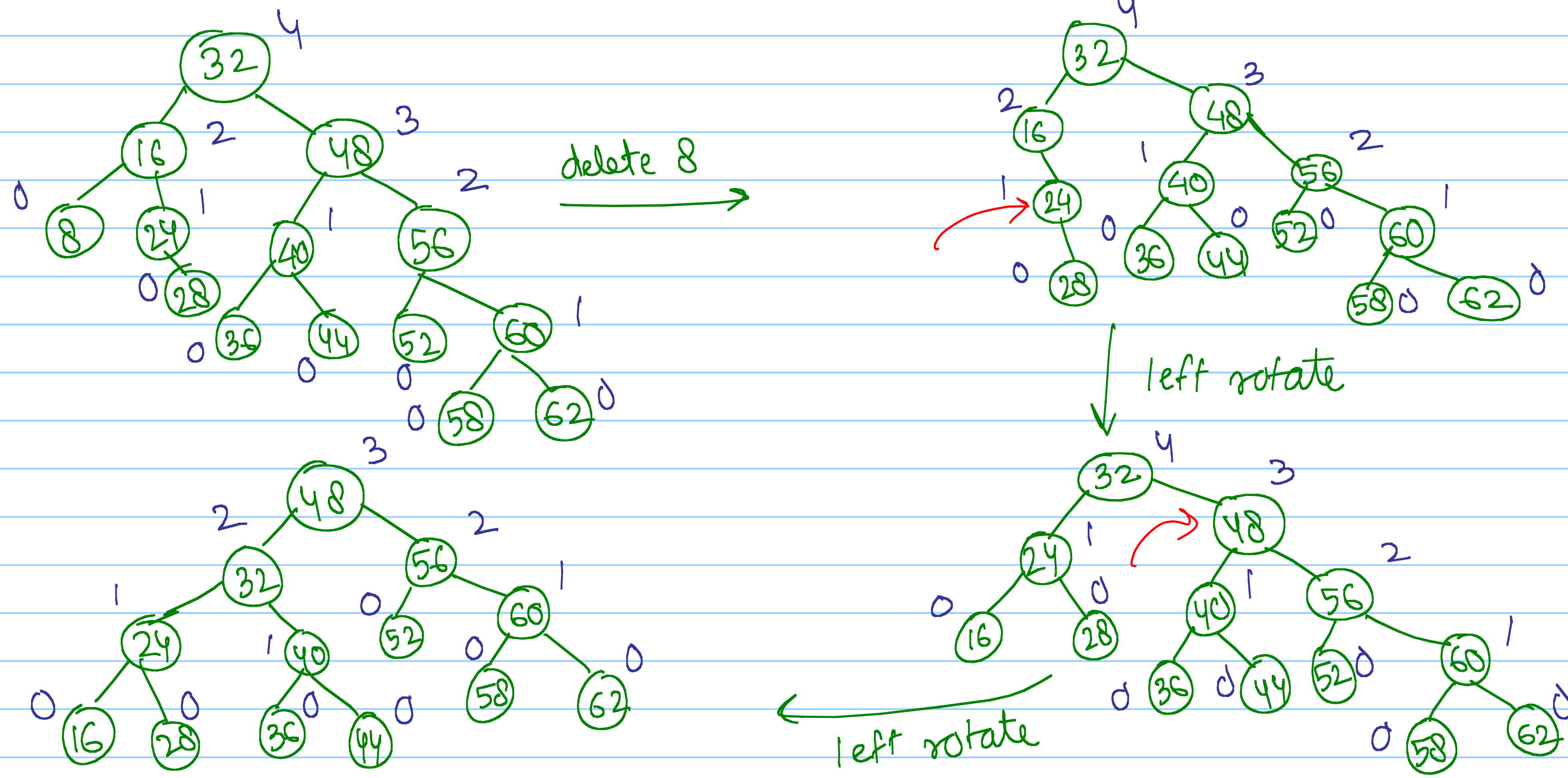
INSERTION AND DELETION

Similar as in BST, followed by rotations to correct imbalances

* INSERTION — one rotation is sufficient (why?) height of root of subtree remains k+1 after rotation
* DELETION — need to check AVL invariant from first point of discrepancy to the root.

O(log n)

Deletion From AVL Trees Excercise



finally AVL !!