2018-09-25 AVL Trees

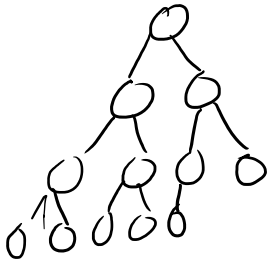
Monday, September 24, 2018 7:17 PM

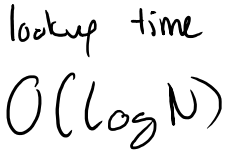
Visualization Resources

- BST Visualization: http://www.cs.usfca.edu/~galles/visualization/BST.html
- AVL Visualization: http://www.cs.usfca.edu/~galles/visualization/AVLtree.html
- Visualization homepage: http://www.cs.usfca.edu/~galles/visualization/Algorithms.html

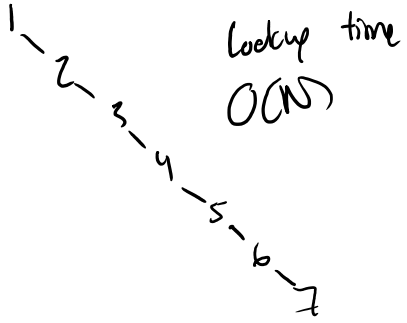
AVL Tree Properties

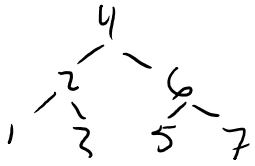
- An AVL tree is a BST that has one additional rule:
  - For each node, the difference between the height of right subtree and left subtree cannot be greater than one
- Unlike BST, AVL trees are guaranteed to be balanced.
- In an ideal world, our BST would look like:





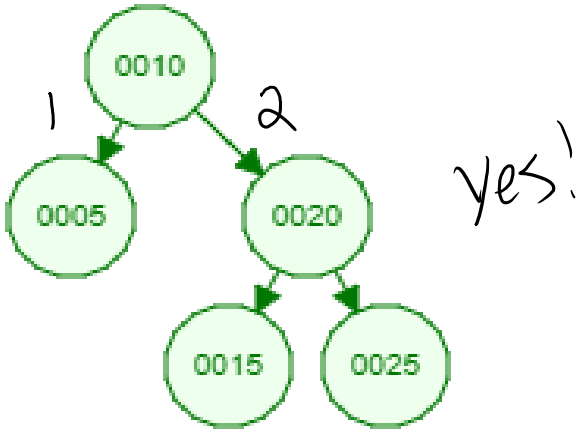
- However, how a BST is constructed affects the BSTs structure. E.g. insert 1, 2, 3, 4, 5, 6, 7 vs insert 4, 6, 2, 1 ,3, 7, 5



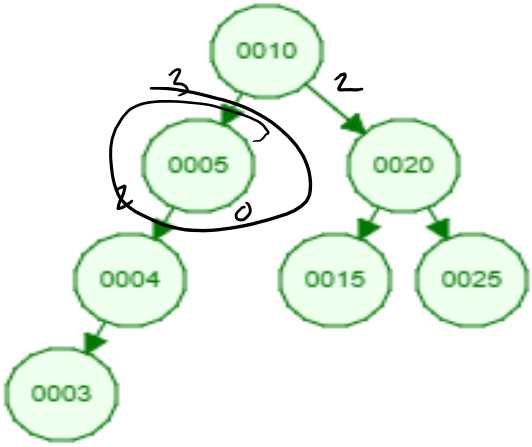


- The AVL tree definition says that the tree must be balanced. AVL trees have a rule that automatically balances the tree on every insert. Thus, on an AVL tree order of insert does not matter. Therefore, AVL trees guarantee LogN behavior.

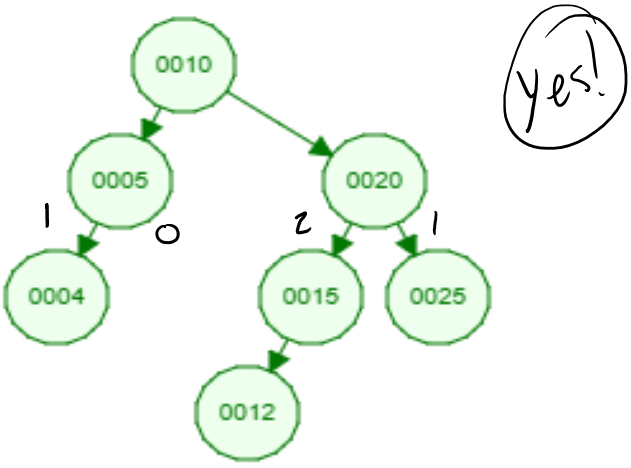
Examples



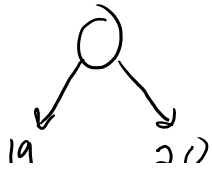
- This is an AVL tree?

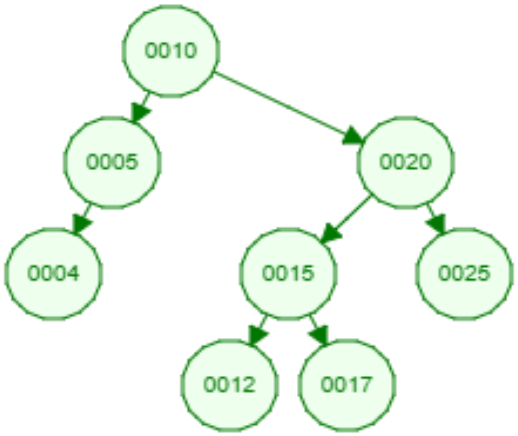


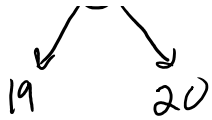
- This is not an AVL tree (why?)



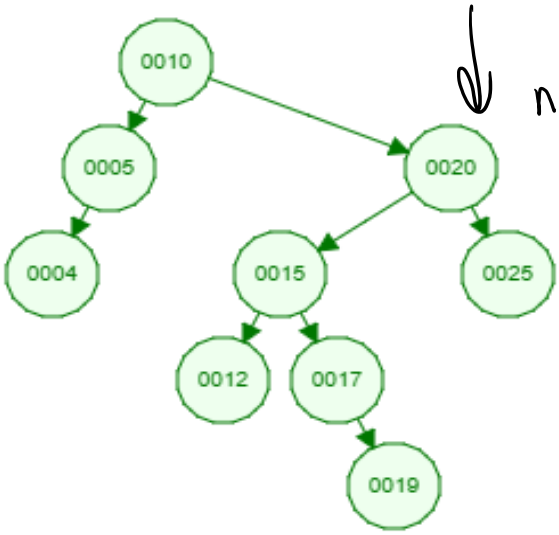
- Is this an AVL tree?

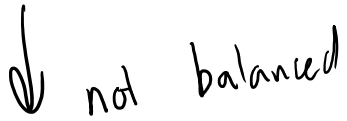






- Is this an AVL tree?

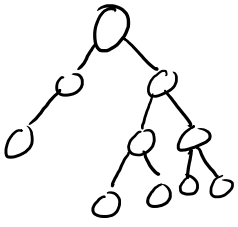




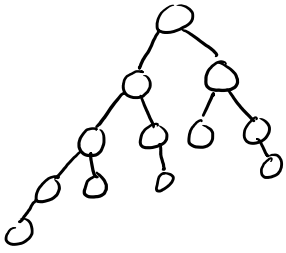
We balance from the bottom up. Thus, 20 is the first node not balanced in our tree.

- Is this an AVL tree?

Draw the most unbalanced tree of height 3 that still AVL compliant



Draw an AVL tree of height 4 that has the fewest nodes possible



Converting a non-AVL tree into an AVL tree using simple (i.e. single) rotations

- Balance factor = right height - left height

Left (counter-clockwise) Rotation

- Occurs when the balance factor is greater than 1 (more nodes on the right)

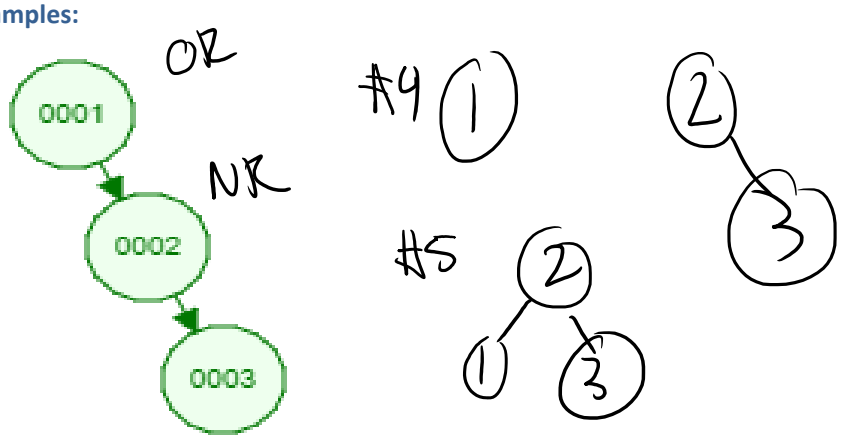
1. At the node whose left and right height differ by more than one, do the following
2. Let OriginalRoot = the node identified in step #1
3. Let NewRoot = OriginalRoot->get**Right**Child()
4. Set OriginalRoot **right** child = NewRoot's **left** child
5. Set NewRoot's **left** child = OriginalRoot

Right (clockwise) Rotation

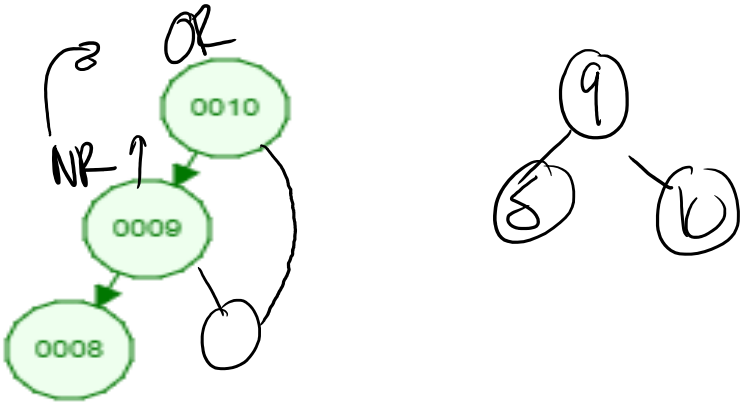
- Occurs when the balance factor is less than 1 (more nodes on the left)

1. At the node whose left and right height differ by more than one, do the following
2. Let OriginalRoot = the node identified in step #1
3. Let NewRoot = OriginalRoot's **left** child
4. Set OriginalRoot's **left** child = NewRoot's **right** child
5. Set NewRoot's **right** child = OriginalRoot

Rotation Examples:

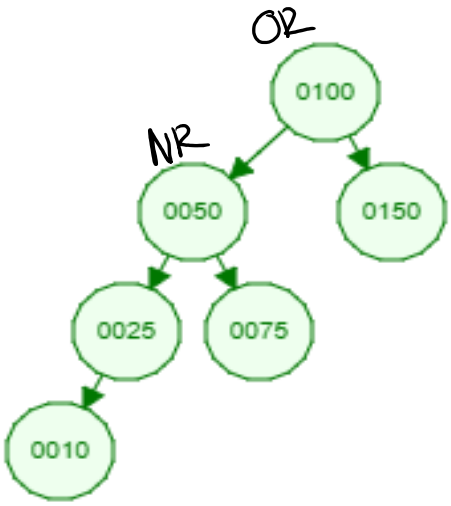


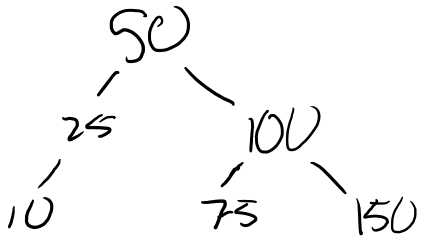
- What is the balance factor?
- What is the type of rotation needed?
- What is the final result?



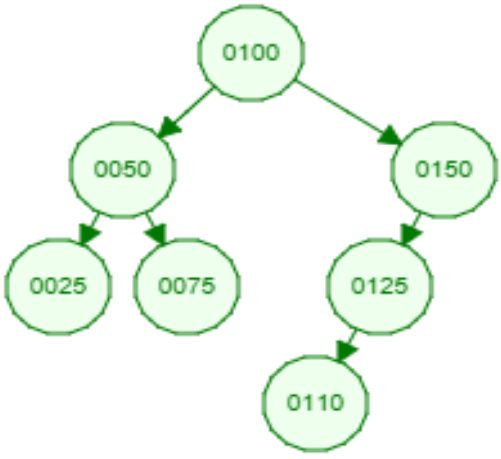
- What is the type of rotation needed?
- What is the final result?

- What is the final result?

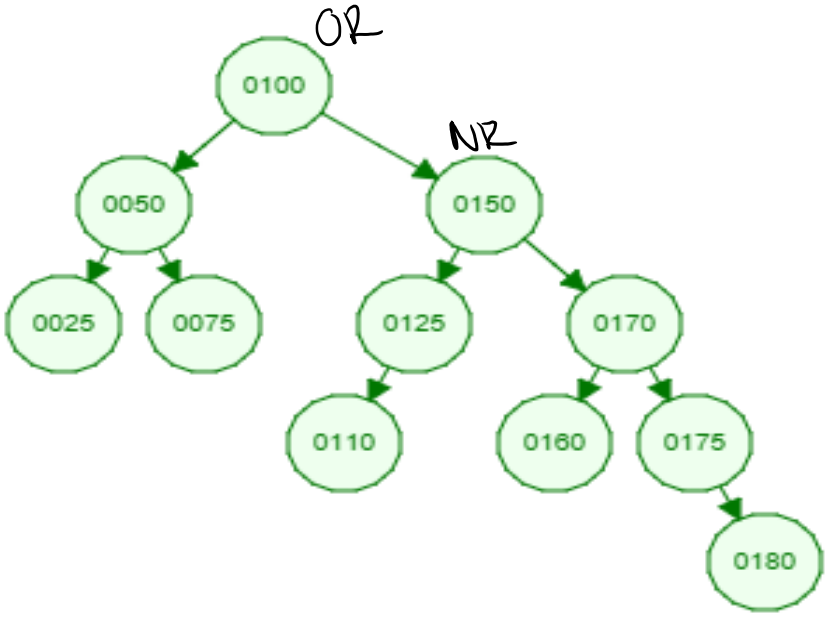


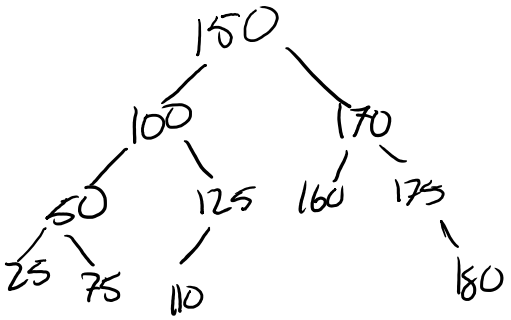


- What is the type of rotation needed?
- What is the final result?



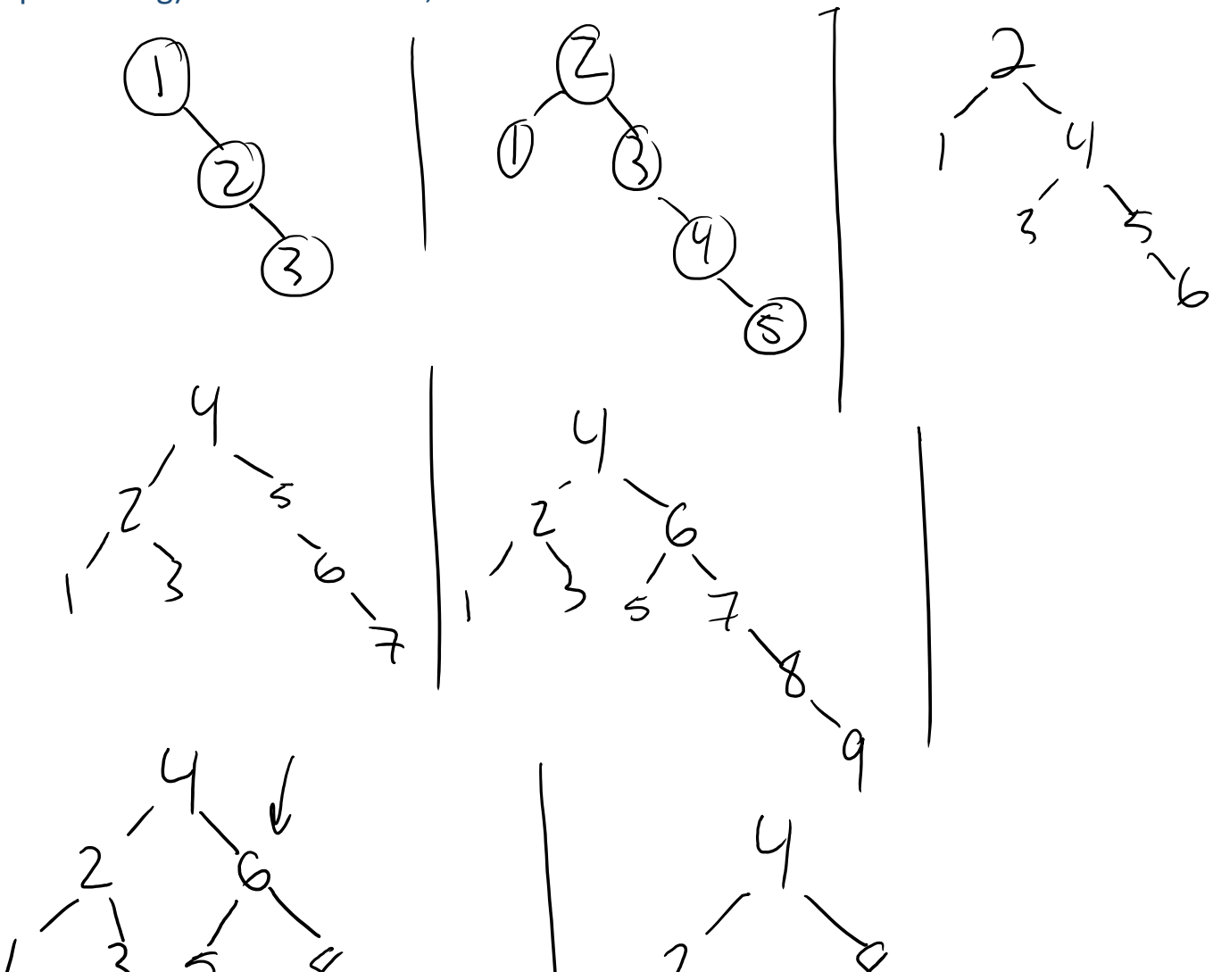
- What is the type of rotation needed?
- What is the final result?

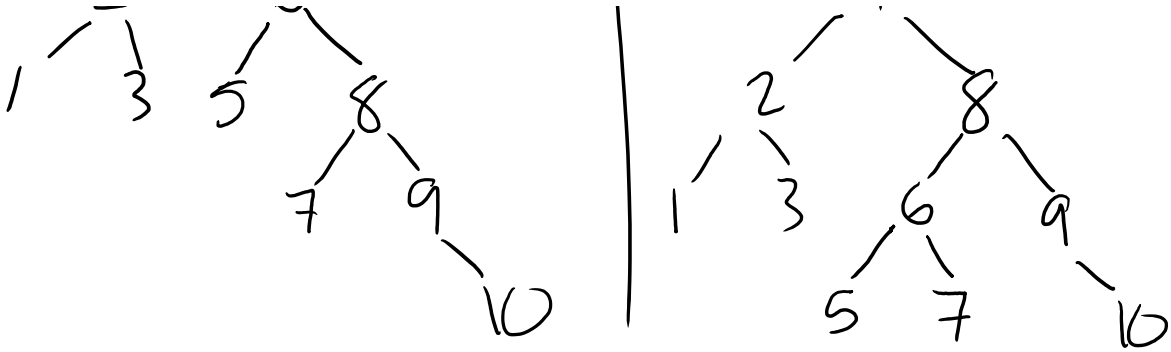




- What is the type of rotation needed?
- What is the final result?

(time permitting) Add values 1-10, remove 10-1




1
3
5
8
7
9
10
2
8
1
3
6
9
5
7
10