

AVL Trees

Defn: A binary tree is a height-balanced p-tree if for each node in the binary tree, the difference in the height of the left and right subtrees is at most p.

Defn: An AVL (Adelson-Velskii, Landis) tree is a binary search height-balanced 1-tree.

Defn: The balance factor of a node, $BF(\text{node})$, in a binary tree is the difference of the left and right subtrees, $hL - hR$.

For any node in an AVL tree, the balance factor is either -1, 0, or 1.

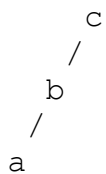
After inserting a new value into an AVL tree, if any node has a BF other than -1, 0, or 1, the AVL tree must be rebalanced.

The AVL tree is rebalanced at the closest ancestor, of the inserted node, that has a BF of -2 or +2. We will call the closest ancestor of the inserted node the pivot node, P.

Four basic rotations are possible where two are single rotations and two are double rotations. Mastering these rotations makes the balancing of an AVL tree pretty easy!!!

Rotation #1: LL (the new node is inserted in the left subtree of the left subtree of the pivot node)

Figure 1 (c is the pivot node)

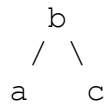


In the the case of an LL, a single rotation right around the pivot node is needed to balance the tree.

Algorithm:

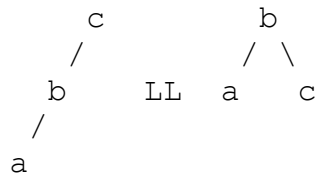
1. Rotate right so that b becomes the new root
2. The leftChild of c points to the rightChild of b
3. The rightChild of b points to c

Figure 2



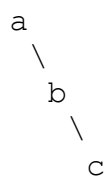
When rebalancing a tree we will show the unbalanced tree followed by the type of rebalance necessary followed by the rebalanced AVL tree. Here is an example.

Figure 3



Rotation #2: RR (the new node is inserted in the right subtree of the right subtree of the pivot node)

Figure 4



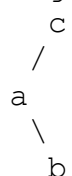
In the the case of an RR, a single rotation left around the pivot node is needed to balance the tree.

Algorithm:

1. Rotate left so that b becomes the new root
2. The rightChild of a points to the leftChild of b
3. The leftChild of b points to a

Rotation #3: LR (the new node is inserted in the right subtree of the left subtree of the pivot node)

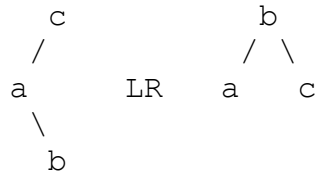
Figure 5



Algorithm:

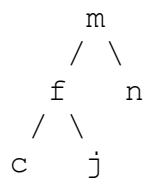
1. Rotate left around a
2. Rotate right around c

Figure 6



P#1: Consider the following AVL tree.

Figure 7

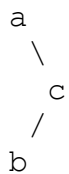


- a) Does this tree need any kind of rebalancing? If so, do it.
- b) Insert g into the tree.
- c) Rebalance if necessary.

P#2: Rework P#1 inserting a into the tree.

Rotation #4: RL (the new node is inserted in the left subtree of the right subtree of the pivot node)

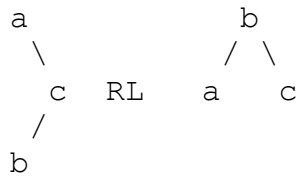
Figure 8



Algorithm:

1. Rotate right around c
2. Rotate left around a

Figure 9



P#3: Insert the following months into an AVL tree. For each rebalancing necessary, show the rotation needed and redraw the balanced tree.

Insert mar, may, nov, aug, apr, jan, dec, jul, feb, jun, oct, sep