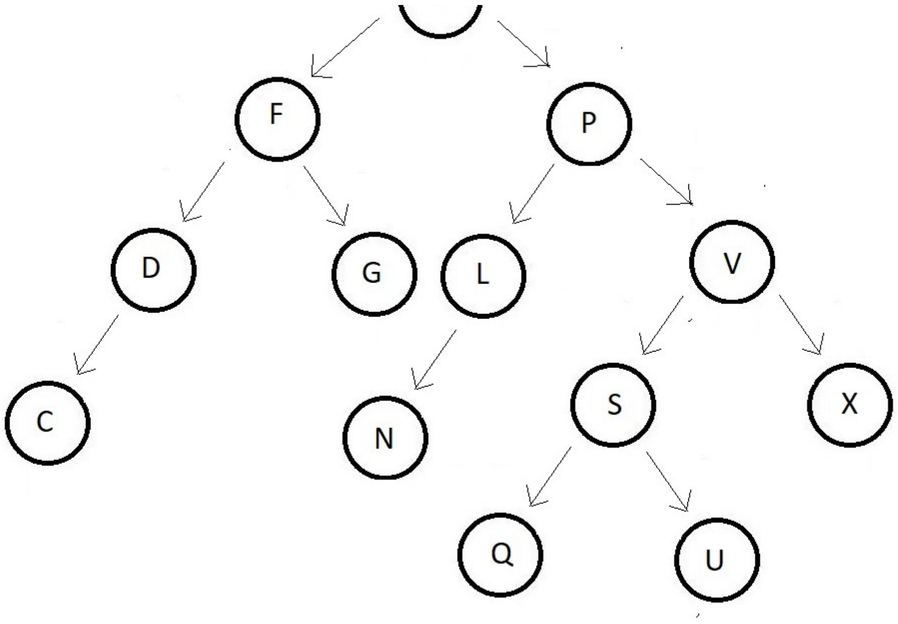
Insertion and Rotation in AVL Tree

In the last lecture, we got introduced to a new variant of the binary search tree, called the AVL trees. We saw why we needed AVL trees and how we judge whether a binary search tree is an AVL tree or not. Today, we'll learn about the rotations in an AVL tree, and why rotation is needed when you insert a new element in an AVL tree. Before we see insertion and rotations in AVL trees, we'll give ourselves a quick brief revision of the characteristics of an AVL tree. These are:

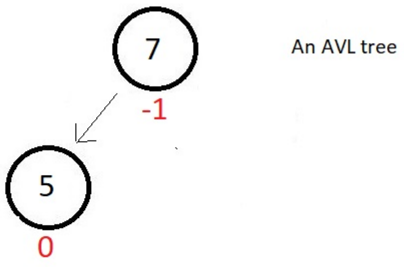
1. AVL trees are height-balanced binary search trees.
2. It gives an upper bound of O(logn) to all the operations possible in a Binary Search Tree.
3. Balance factor, BF = Height of Right Subtree - Height of Left Subtree, and for a binary search tree to be an AVL tree, |BF| should be less than or equal to 1 for all of its nodes.

An example of an AVL tree is:

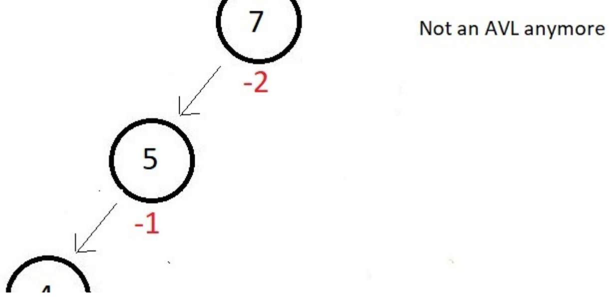


Now, before we proceed to see what different types of rotation in an AVL tree we have, we would first like to know why rotation is even done.

In an AVL tree, rotations are performed to avoid the unbalancing of a node caused by insertion. Now, suppose we have a small AVL tree having just these two nodes.



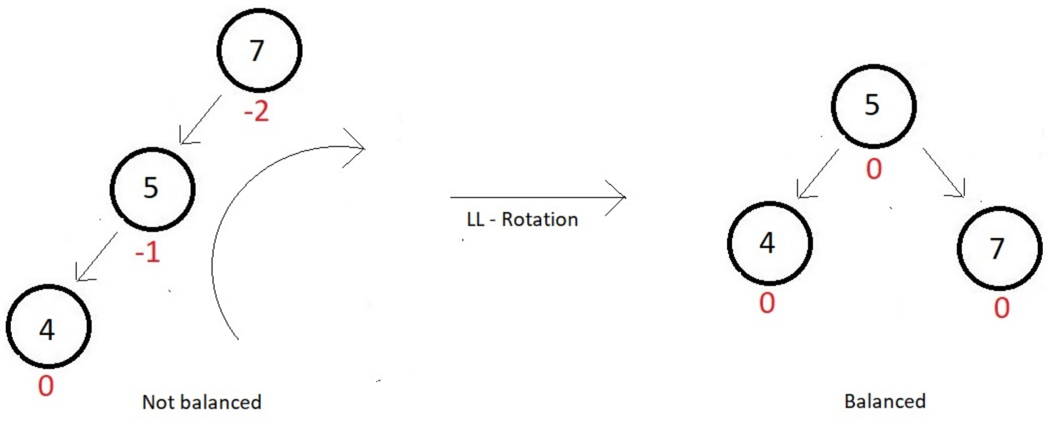
And you can see, the balance factor of both the nodes are good, but as soon as we insert a new node having data 4, our updated tree becomes unbalanced to the left. The absolute balance factor of node 7 becomes greater than 1.



So, the method you will follow to make this tree an AVL again is called **rotation**. Now, rotations can be of different types, one of them being the **LL rotation**.

LL Rotation:

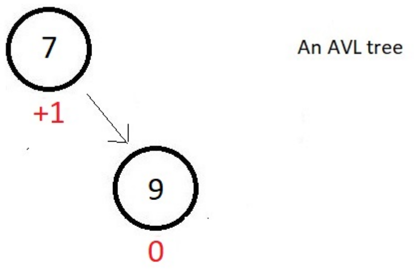
The name LL, just because we inserted the new element to the left subtree of the root. In this rotation technique, you just simply rotate your tree one time in the clockwise direction as shown below.



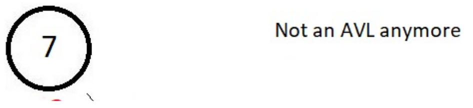
So, our tree got balanced again, with a perfect balance factor at each of its nodes. Next, we have the RR rotation.

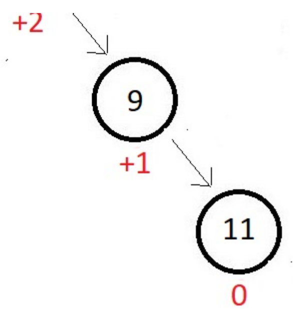
RR Rotation:

Now, suppose we have a small AVL tree having just these two nodes.

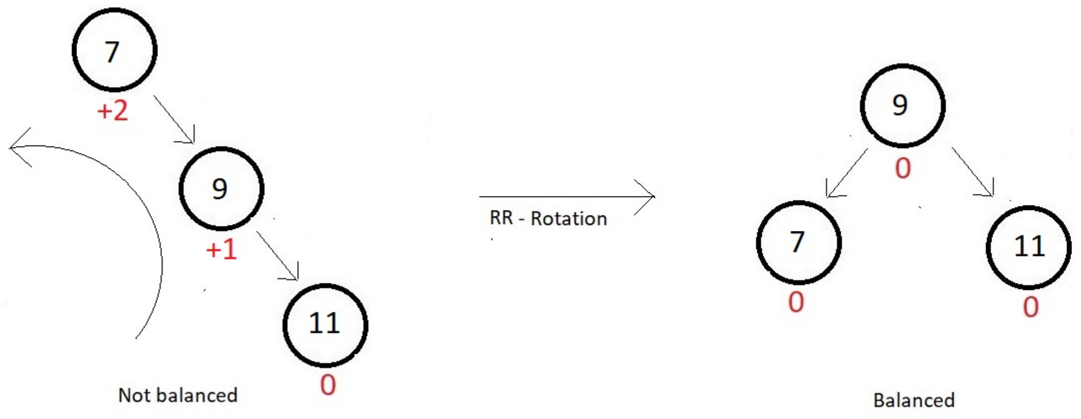


And you can see, the balance factor of both the nodes are good, but as soon as we insert a new node having data 11, our updated tree becomes unbalanced to the right. The absolute balance factor of node 7 becomes greater than 1.





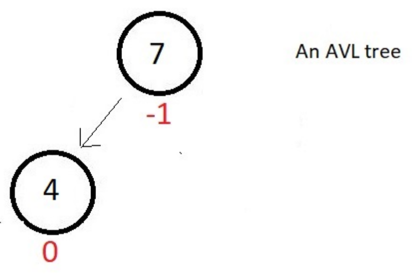
So, the method you will follow now to make this tree an AVL again is called the **RR rotation**. The name RR, just because we inserted the new element to the right subtree of the root. In this rotation technique, you just simply rotate your tree one time in an anti-clockwise direction as shown below.



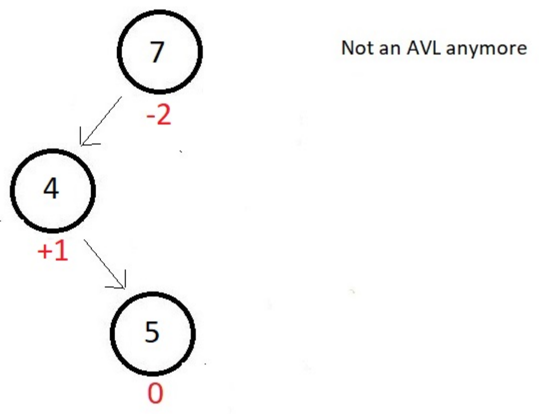
So, our tree got balanced again, with a perfect balance factor at each of its nodes. Next, we have the LR rotation.

LR Rotation:

Now, suppose we have a small AVL tree having just these two nodes.

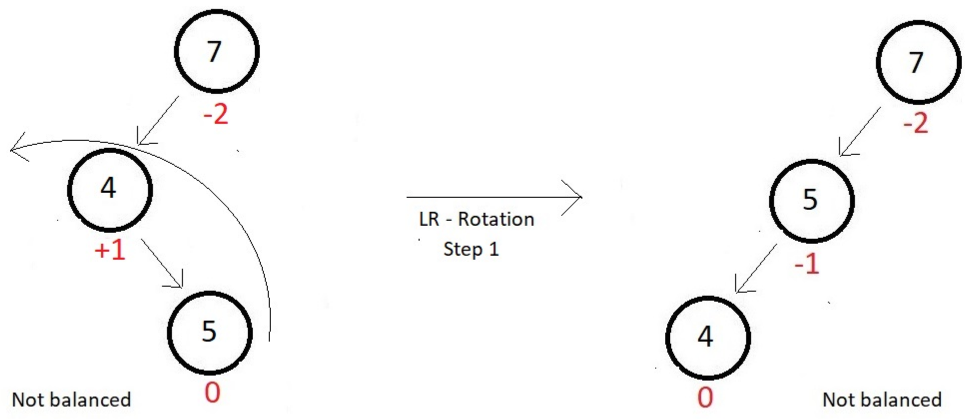


And you can see, the balance factor of both the nodes are good, but as soon as we insert a new node having data 5, our updated tree becomes unbalanced to the left. The absolute balance factor of node 7 becomes greater than 1.

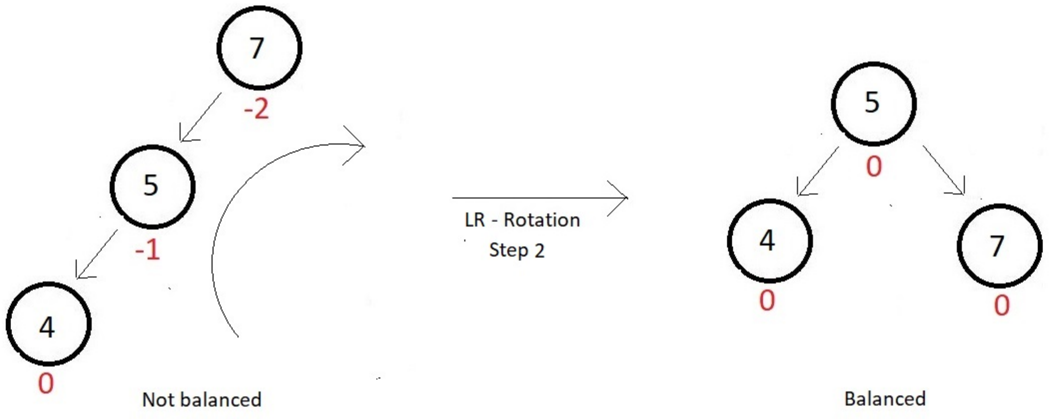


So, the method you will follow now to make this tree an AVL again is called the **LR rotation**. The name LR, just because we inserted the new element to the right to the left subtree of the root. In this rotation technique, there is a subtle complexity, which says, first rotate the left subtree in the anticlockwise direction, and then the whole tree in the clockwise direction. Follow the two steps illustrated below:

Step 1:



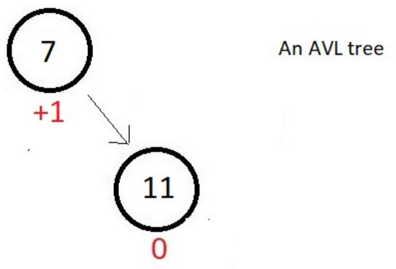
Step 2:



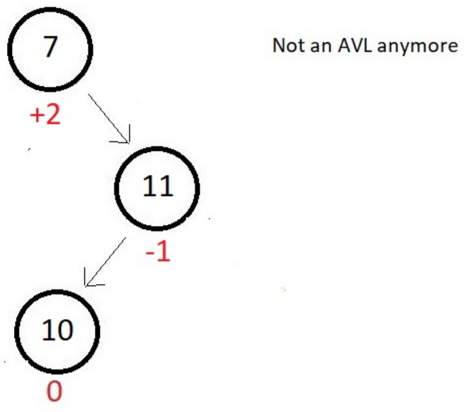
So, our tree got balanced again, with a perfect balance factor at each of its nodes. Although it was a bit clumsy, it was achievable. Next, we have the RL rotation.

RL Rotation:

Now, suppose we have a small AVL tree having just these two nodes.

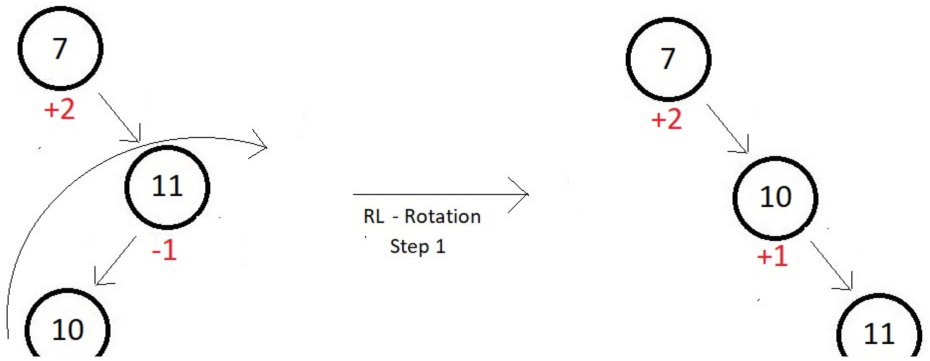


And you can see, the balance factor of both the nodes are good, but as soon as we insert a new node having data 10, our updated tree becomes unbalanced to the right. The absolute balance factor of node 7 becomes greater than 1.

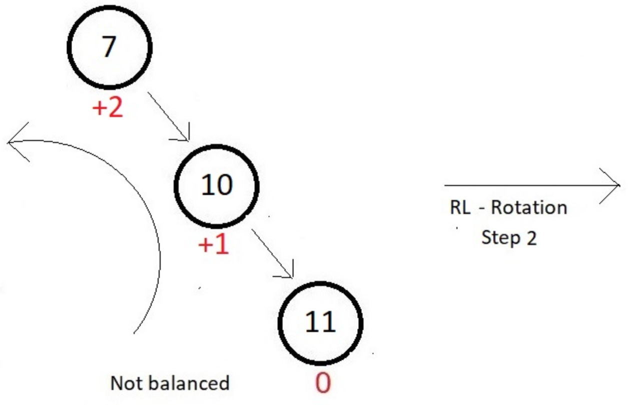


So, the method you will follow now to make this tree an AVL again is called the **RL rotation**. The name RL, just because we inserted the new element to the left to the right subtree of the root. We follow the same technique we used above, which says, first rotate the right subtree in the clockwise direction, and then the whole tree in the anticlockwise direction. Follow the two steps illustrated below:

Step 1:



Step 2:



So, our tree got balanced again, with a perfect balance factor at each of its nodes. And those were our different types of rotation techniques that helped gain the balance of our AVL trees back after the insertion of new elements.
You all might be wondering what would have happened if the trees we selected to demonstrate the types of rotation had been complex. Well, to understand those complex problems, knowing the basics was important. That's why we choose the smallest possible AVL trees. No doubt, we'll follow those complexities in our next video. Stay tuned.

I appreciate your support throughout. I hope you enjoyed the tutorial. If you genuinely appreciate my work, please let your friends know about this course too. If you haven't checked out the whole playlist yet, move on to codewithharry.com or my YouTube channel to access it. See you all in the next tutorial where we'll see these different types of rotations in a few complex situations. Till then keep coding.

Previous Next

CodeWithHarry