Binary Search Tree 101

All left child of a parent must have values less than the parent.

All right child of a parent must have values more than the parent.

AVL Tree is a height-balanced BST – The heights of left and right subtrees of every node differ by at most one.

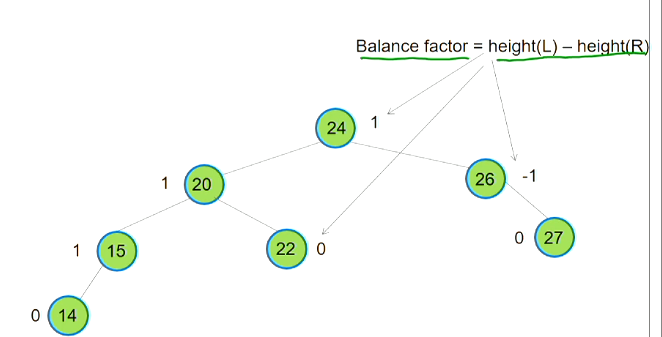
If at any time they differ by more than one, rebalancing is done to restore this property.

If a tree is height balanced, then we know that the height of the tree cannot be more than O(log n). This means that for all search, insertion, and deletion, the time complexity will all be O(log n).

insertion is a bit costlier as we want to make the height balanced every time after an insertion or deletion.

balance factor = height(L) – height(R)

If balance factor is not 1, 0, or -1, then it is not an AVL Tree.



Balancing AVL Tree

To balance an AVL tree after deletion/insertion we have four possible scenarios:

* Left-left
* Left-right
* Right-right
* Right-left

Left-left occurs when a node has a balance factor of +2 and its left child has a balance factor of 0 or more.

Idea:

Consider the 3 nodes and the subtrees connecting the 3 nodes.

Build a triangle such that the left and right child conditions are satisfied.

Attach the subtrees back into the triangle from left to right.