AVL Trees:

it is Balanced Binary Search Tree

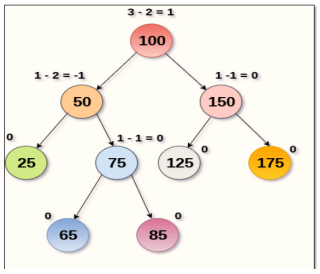
BST has skewness issue, which leads to O(n) search time instead of ideal O(logn) time

while inserting a element into BST it can be balanced if we pick median out of existing elements

(but takes extra O(nlogn) to sort)

Properties of AVL Trees

- It is a binary search tree, and
- For any node X, the height of left subtree of X and height of right subtree of X differ by *at most 1*.
- i.e. An AVL tree is a height balanced BST !

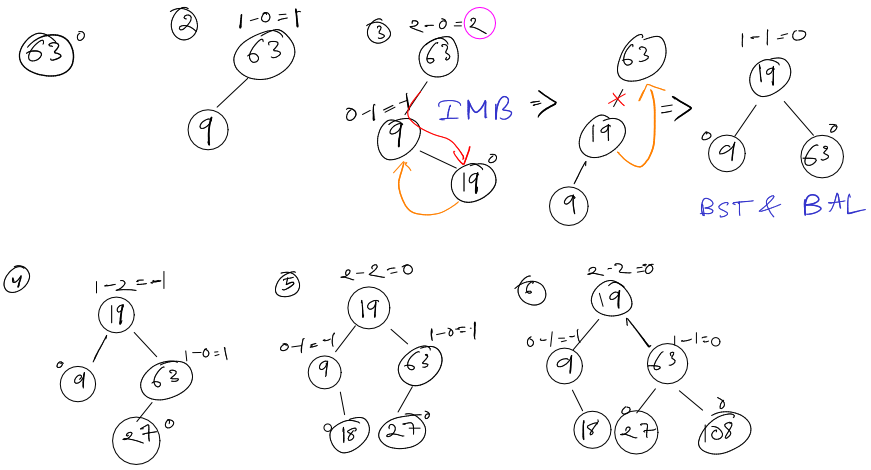


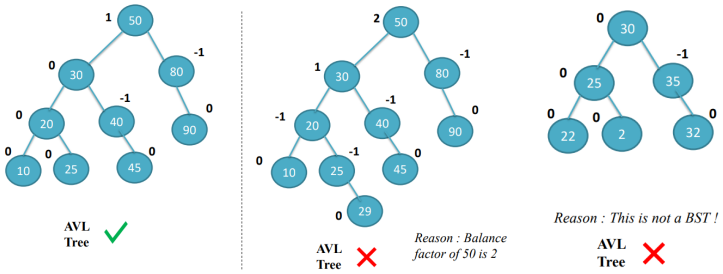
balance factor = height of left subtree − height of right subtree

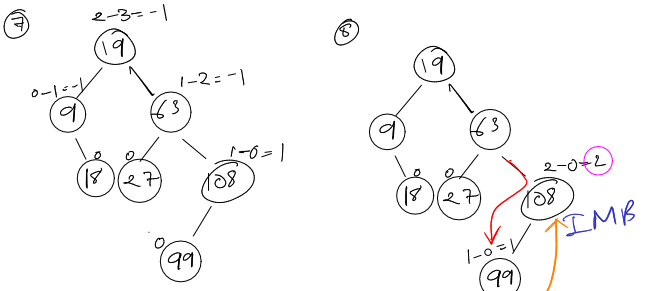
It should be -1 or 0 or 1 none else

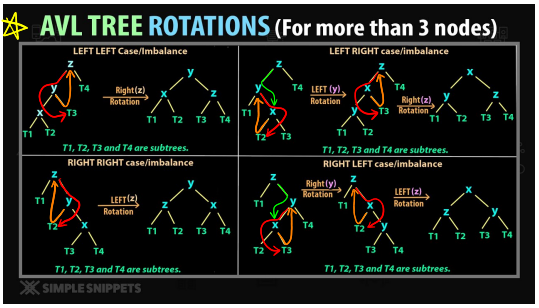
Exercise:

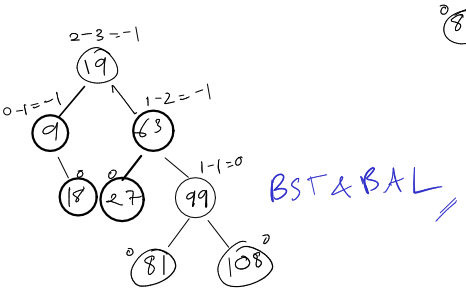
Construct the AVL tree for : 63, 9, 19, 27, 18, 108, 99, 81











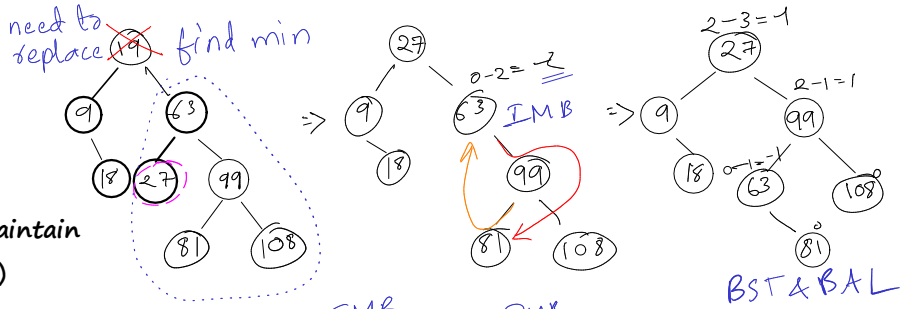
While Inserting elements in BST manner,

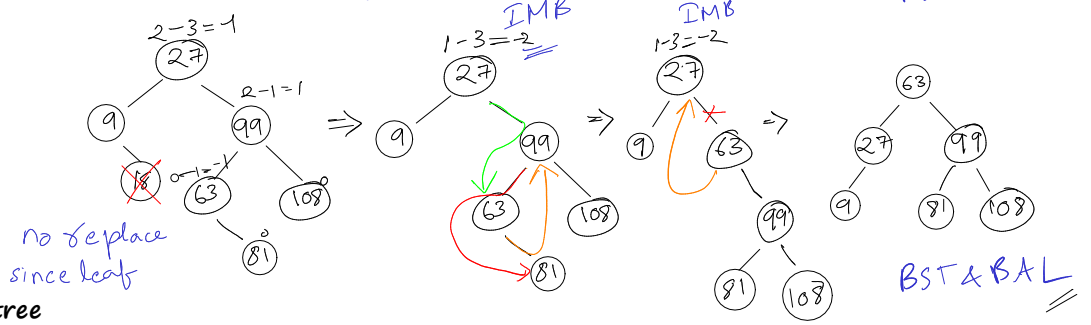
1. build complete BST (balanced or unbalanced) , once built then traverse and rebalance
2. at each insertion check for imbalance and rebalance if any for every insertion

For Deletion use BST delete accordingly then rebalance if any imbalance

For Insertion / Deletion in AVL Tree need to maintain *BST property & Balance Factor (-1 or 0 or 1)

Deletion:





links:

https://en.wikipedia.org/wiki/AVL_tree

https://iq.opengenus.org/avl_tree/