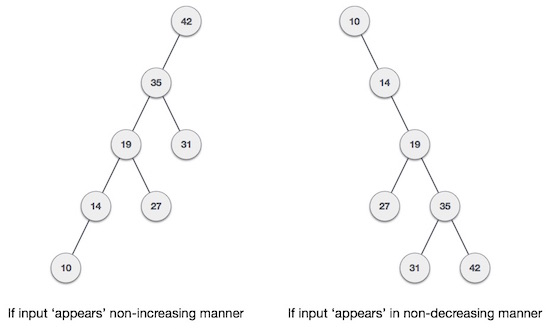
Top of Form

**TREES (Contd..)**

**AVL TREE**

AVL tree is a self-balancing Binary Search Tree (BST) where the difference between heights of left and right subtrees cannot be more than one for all nodes.

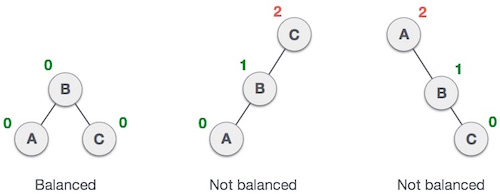
What if the input to binary search tree comes in a sorted (ascending or descending) manner? It will then look like this −



It is observed that BST's worst-case performance is closest to linear search algorithms, that is Ο(n). In real-time data, we cannot predict data pattern and their frequencies. So, a need arises to balance out the existing BST.

Named after their inventor Adelson, Velski & Landis, AVL trees are height balancing binary search tree. AVL tree checks the height of the left and the right sub-trees and assures that the difference is not more than 1. This difference is called the Balance Factor.

Here we see that the first tree is balanced and the next two trees are not balanced −



In the second tree, the left subtree of C has height 2 and the right subtree has height 0, so the difference is 2. In the third tree, the right subtree of A has height 2 and the left is missing, so it is 0, and the difference is 2 again. AVL tree permits difference (balance factor) to be only 1.

***BalanceFactor*** = height(left-sutree) − height(right-sutree)

If the difference in the height of left and right sub-trees is more than 1, the tree is balanced using some rotation techniques.

**AVL Rotations**

To balance itself, an AVL tree may perform the following four kinds of rotations −

* Left rotation
* Right rotation
* Left-Right rotation
* Right-Left rotation

The first two rotations are single rotations and the next two rotations are double rotations. To have an unbalanced tree, we at least need a tree of height 2. With this simple tree, let's understand them one by one.

**Left Rotation**

If a tree becomes unbalanced, when a node is inserted into the right subtree of the right subtree, then we perform a single left rotation −



In our example, node A has become unbalanced as a node is inserted in the right subtree of A's right subtree. We perform the left rotation by making A the left-subtree of B.

**Right Rotation**

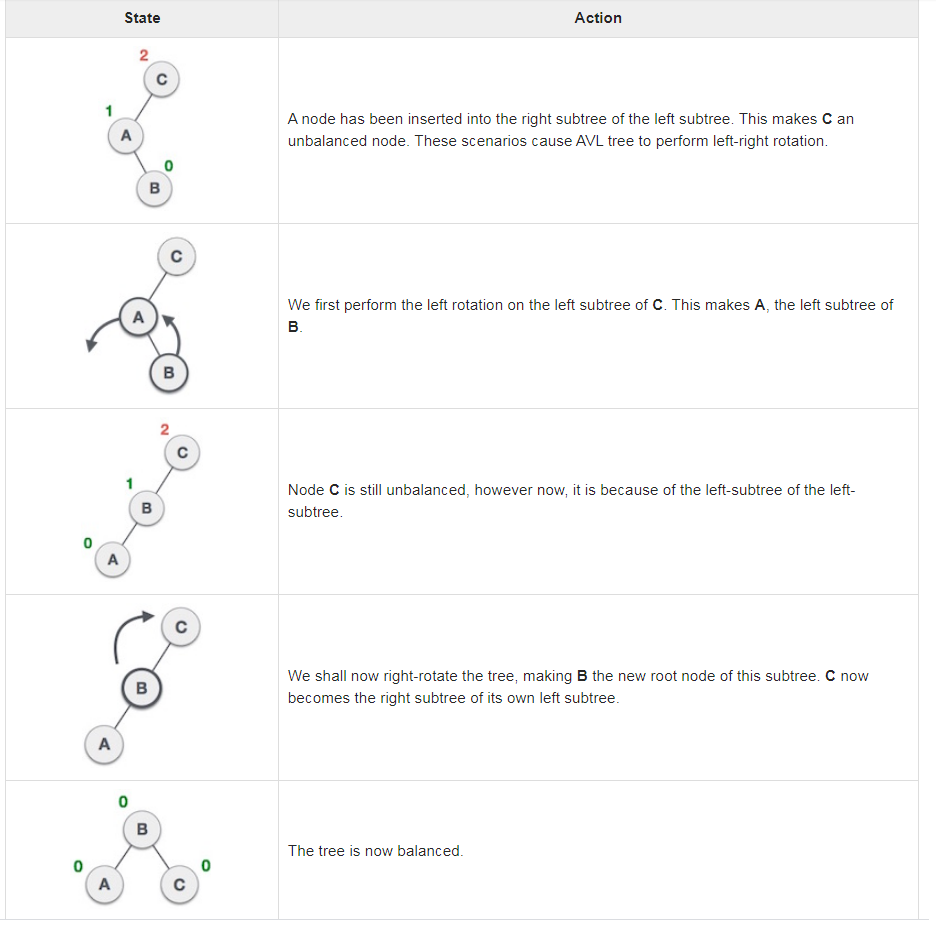
AVL tree may become unbalanced, if a node is inserted in the left subtree of the left subtree. The tree then needs a right rotation.



As depicted, the unbalanced node becomes the right child of its left child by performing a right rotation.

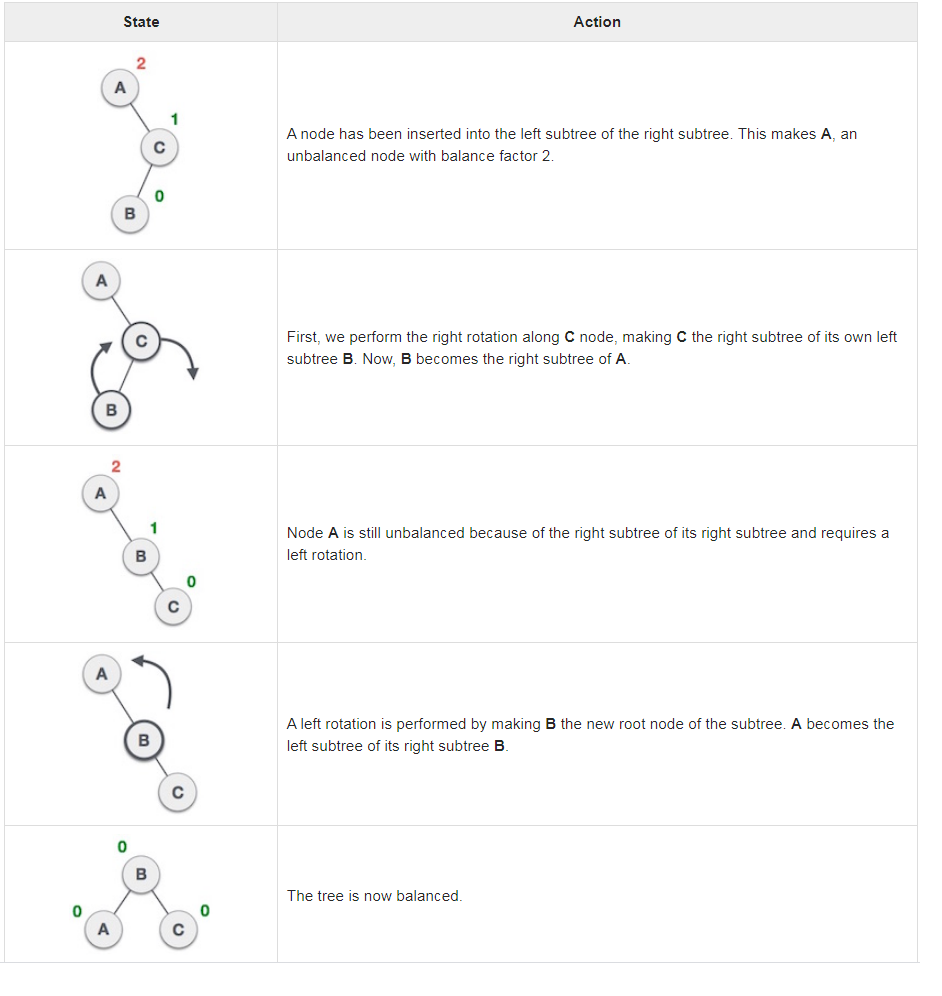
**Left-Right Rotation**

Double rotations are slightly complex version of already explained versions of rotations. To understand them better, we should take note of each action performed while rotation. Let's first check how to perform Left-Right rotation. A left-right rotation is a combination of left rotation followed by right rotation.



#### Right-Left Rotation

The second type of double rotation is Right-Left Rotation. It is a combination of right rotation followed by left rotation.



**Red Black Tree:**

Red black tree is a self balanced binary search tree invented by Rudolf Bayer in 1972. It is also called as 'symmetric binary B tree'. It has good worst case running time for its operations. It performs all the operations such as searching, insertion, deletion, in O(log n ) time where n is the number of nodes in the tree. In Red black tree, no data is stored at the leaves.

**Properties of Red Black Tree**

1. The color of a node is either black or red.
2. The color of root node is always black.
3. All leaf nodes are of black color.
4. Every red node has both children of black color.
5. Every path from a node to its leaf nodes contains equal number of black nodes.
6. The longest path from the root node to any leaf node is no more than twice as long as the shortest path from the root to any other leaf in that tree.

**RELEVANT READING MATERIAL AND REFERENCES:**

**Source Notes:**

1. <https://www.tutorialspoint.com/data_structures_algorithms/avl_tree_algorithm.htm>
2. <https://www.javatpoint.com/red-black-tree>

**Lecture Video:**

1. <https://www.youtube.com/watch?v=-ODwekYuDkM>
2. <https://www.youtube.com/watch?v=3RQtq7PDHog>
3. <https://www.youtube.com/watch?v=qA02XWRTBdw>

**Online Notes:**

1. <http://www.vssut.ac.in/lecture_notes/lecture1428550942.pdf>

**Text Book Reading:**

1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.
2. Lipschutz, S., “*Data Structures, Schaum's Outline Series*”, Tata McGraw Hill.

**Online Book Reference:**

1. <https://www.edutechlearners.com/download/books/DS.pdf>

**In addition: PPT can be also be given.**