AVL Tree Data Structure

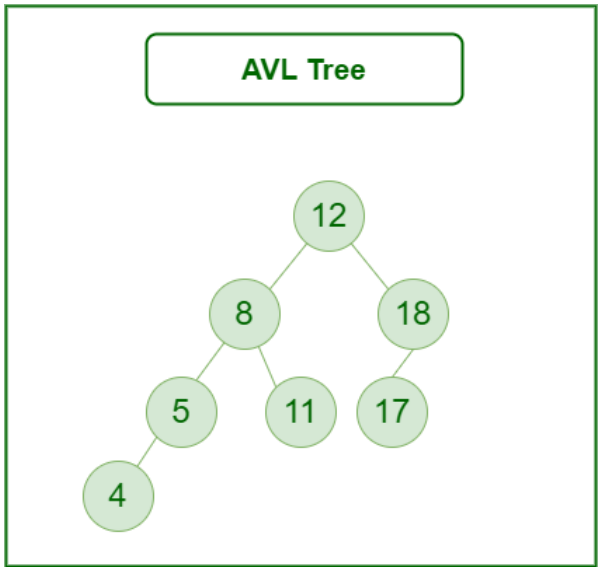
Last Updated : 24 Feb, 2025

An **AVL tree** defined as a self-balancing Binary Search Tree (BST) where the difference between heights of left and right subtrees for any node cannot be more than one.

- The absolute difference between the heights of the left subtree and the right subtree for any node is known as the **balance factor** of the node. The balance factor for all nodes must be less than or equal to 1.
- Every AVL tree is also a Binary Search Tree (Left subtree values Smaller and Right subtree values grater for every node), but every BST is not AVL Tree. For example, the second diagram below is not an AVL Tree.
- The main advantage of an AVL Tree is, the time complexities of all operations (search, insert and delete, max, min, floor and ceiling) become O(Log n). This happens because height of an AVL tree is bounded by O(Log n). In case of a normal BST, the height can go up to O(n).
- An AVL tree maintains its height by doing some extra work during insert and delete operations. It mainly uses rotations to maintain both BST properties and height balance.
- There exist other self-balancing BSTs also like Red Black Tree. Red Black tree is more complex, but used more in practice as it is less restrictive in terms of left and right subtree height differences.

Example of an AVL Tree:

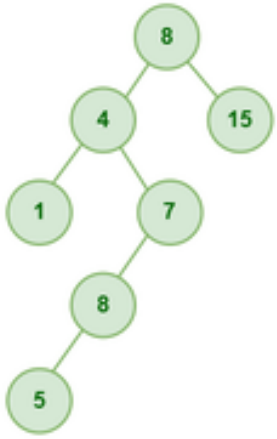
The balance factors for different nodes are : 12 :1, 8:1, 18:1, 5:1, 11:0, 17:0 and 4:0. Since all differences are less than or equal to 1, the tree is an AVL tree.



AVL tree

Example of a BST which is NOT AVL:

The Below Tree is **NOT an AVL Tree** as the balance factor for nodes 8, 4 and 7 is more than 1.



Not an AVL Tree

Operations on an AVL Tree:

- **Searching** : It is same as normal Binary Search Tree (BST) as an AVL Tree is always a BST. So we can use the same implementation as BST. The

- **Insertion** : It does rotations along with normal BST insertion to make sure that the balance factor of the impacted nodes is less than or equal to 1 after insertion
- **Deletion** : It also does rotations along with normal BST deltion to make sure that the balance factor of the impacted nodes is less than or equal to 1 after deletion.

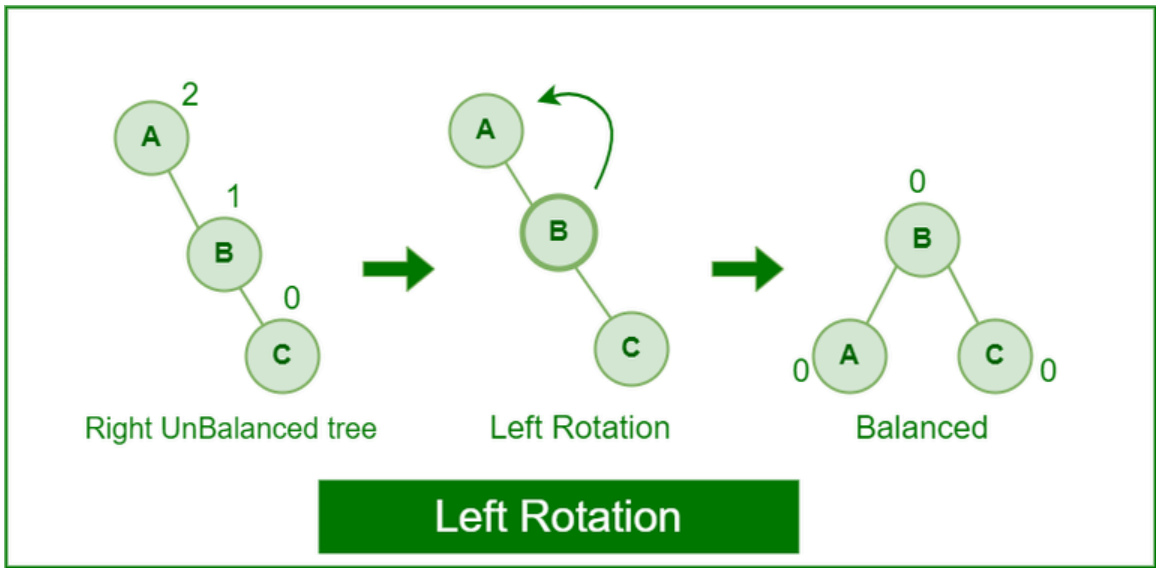
Please refer Insertion in AVL Tree and Deletion in AVL Tree for details.

Rotating the subtrees (Used in Insertion and Deletion)

An AVL tree may rotate in one of the following four ways to keep itself balanced while making sure that the BST properties are maintained.

Left Rotation:

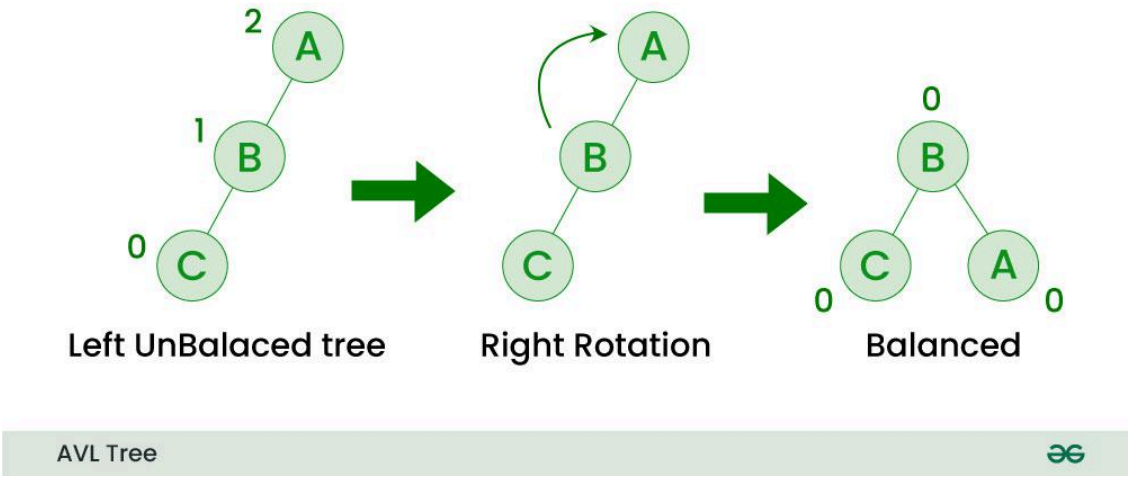
When a node is added into the right subtree of the right subtree, if the tree gets out of balance, we do a single left rotation.



Left-Rotation in AVL tree

Right Rotation:

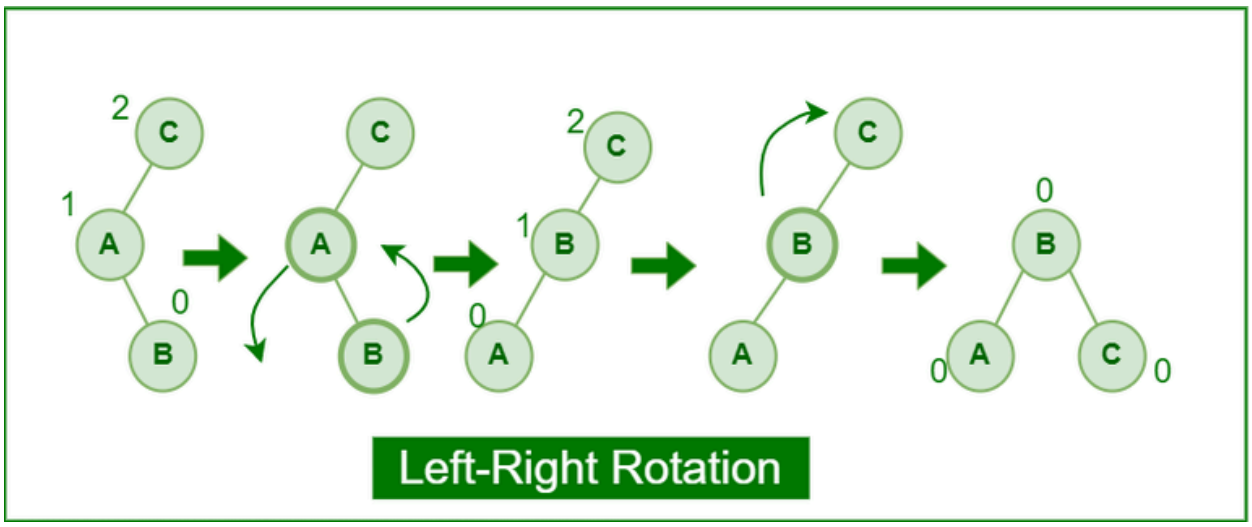
If a node is added to the left subtree of the left subtree, the AVL tree may get out of balance, we do a single right rotation.



Right-Rotation in AVL Tree

Left-Right Rotation:

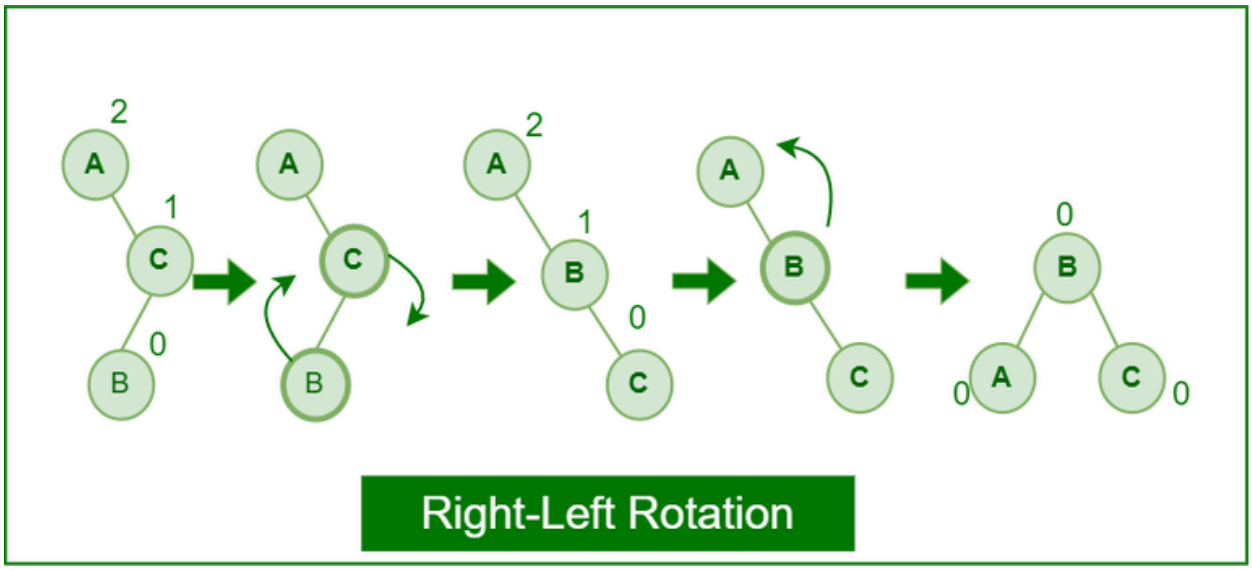
A left-right rotation is a combination in which first left rotation takes place after that right rotation executes.



Left-Right Rotation in AVL tree

Right-Left Rotation:

A right-left rotation is a combination in which first right rotation takes place after that left rotation executes.



Right-Left Rotation in AVL tree

Advantages of AVL Tree:

1. AVL trees can self-balance themselves and therefore provides time complexity as O(Log n) for search, insert and delete.
2. It is a BST only (with balancing), so items can be traversed in sorted order.
3. Since the balancing rules are strict compared to Red Black Tree, AVL trees in general have relatively less height and hence the search is faster.
4. AVL tree is relatively less complex to understand and implement compared to Red Black Trees.

Disadvantages of AVL Tree:

1. It is difficult to implement compared to normal BST and easier compared to Red Black
2. Less used compared to Red-Black trees. Due to its rather strict balance, AVL trees provide complicated insertion and removal operations as more rotations are performed.

Applications of AVL Tree:

1. AVL Tree is used as a first example self balancing BST in teaching DSA as it

2. Applications, where insertions and deletions are less common but frequent data lookups along with other operations of BST like sorted traversal, floor, ceil, min and max.
3. Red Black tree is more commonly implemented in language libraries like map in C++, set in C++, TreeMap in Java and TreeSet in Java.
4. AVL Trees can be used in a real time environment where predictable and consistent performance is required.

Related Articles:

- Insertion in an AVL Tree
- Deletion in an AVL Tree
- Red Black Tree

An AVL tree defined as a self-balancing Binary Search Tree (BST) where the difference between heights of left and right subtrees for any node cannot be more than one. The absolute difference between the heights o...

4 min read

What is AVL Tree | AVL Tree meaning

An AVL is a self-balancing Binary Search Tree (BST) where the difference between the heights of left and right subtrees of any node cannot be more than one. KEY POINTSIt is height balanced treeIt is a binary searc...

2 min read

Insertion in an 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. Example of AVL Tree: The above tree is AVL because the...

15+ min read

Insertion, Searching and Deletion in AVL trees containing a parent node pointer

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. The insertion and deletion in AVL trees have been discussed...

15+ min read

Deletion in an AVL Tree

We have discussed AVL insertion in the previous post. In this post, we will follow a similar approach for deletion. Steps to follow for deletion. To make sure that the given tree remains AVL after every deletion, we...

15+ min read

How is an AVL tree different from a B-tree?

AVL Trees: AVL tree is a self-balancing binary search tree in which each node maintain an extra factor which is called balance factor whose value is either -1, 0 or 1. B-Tree: A B-tree is a self - balancing tree data structure...

1 min read

Practice questions on Height balanced/AVL Tree

AVL tree is binary search tree with additional property that difference between height of left sub-tree and right sub-tree of any node canâ€™t be more than 1. Here are some key points about AVL trees: If there are n...

4 min read

AVL with duplicate keys

Please refer below post before reading about AVL tree handling of duplicates. How to handle duplicates in Binary Search Tree?This is to augment AVL tree node to store count together with regular fields like key, left...

15+ min read

In this article we will see that how to calculate number of elements which are greater than given value in AVL tree. Examples: Input : x = 5 Root of below AVL tree 9 / \ 1 10 / \ \ 0 5 11 / / \ -1 2 6 Output : 4 Explanation:...

15+ min read

Difference between Binary Search Tree and AVL Tree

Binary Search Tree:A binary Search Tree is a node-based binary tree data structure that has the following properties: The left subtree of a node contains only nodes with keys lesser than the nodeâ€™s key.The right...

2 min read