Department of Computer science and Engineering

PES UNIVERSITY

UE19CS202: Data Structures and its Applications (4-0-0-4-4)

AVL TREES

Abstract

Balanced tree, Need for Balanced tree, definition, AVL Trees, Rotation.

Dr.Sandesh and Saritha

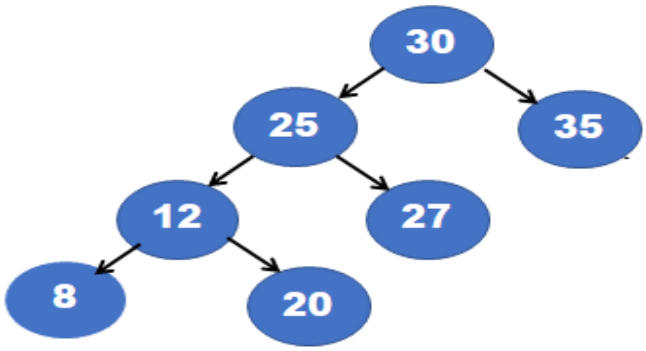
Sandesh_bj@pes.edu

Saritha.k@pes.edu

Balanced tree

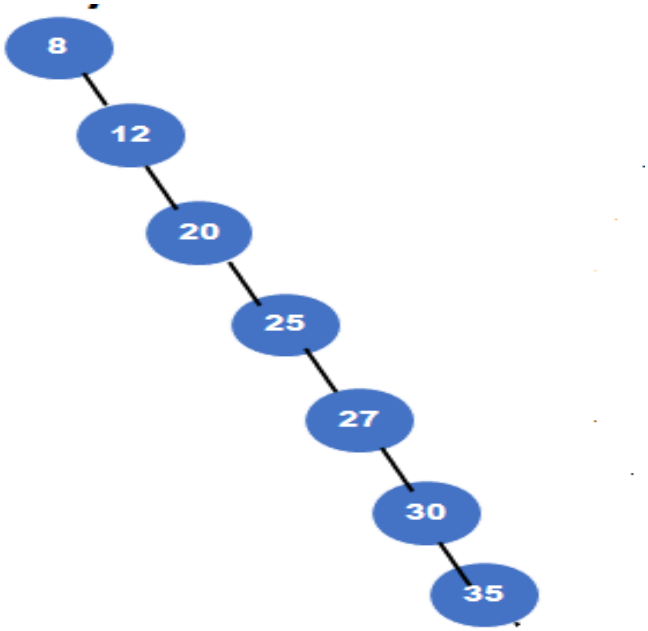
Most operations on a Binary search tree take time directly proportional to the height of the tree, so it is important to keep the height of the tree small.

A balanced binary search tree is a tree that naturally keeps its height small for a sequence of insertion and deletion operation.



Balanced Binary search tree

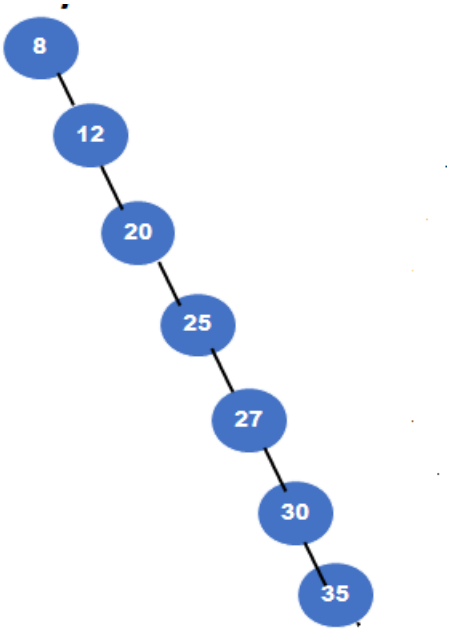
The tree shown above is balanced because the difference between the heights of left subtree and right subtree is not more than one.



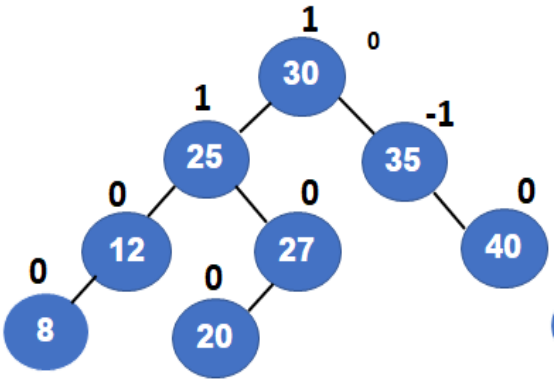
The tree shown above is unbalanced because the tree right side is 6 leaves taller than the left hand side.

Why do Binary search trees have to be balanced?

Balanced search tree decreases the number of comparisons required to find a data element. For example consider a below unbalanced tree, to search for an element 35 in the below tree seven comparisons is required. Whereas in balanced tree it requires only 2 comparisons which means the search performance increased by 50% in a balanced tree.



(i)Unbalanced tree



(ii) Balanced tree

There are different algorithms used to balance the binary search tree such as

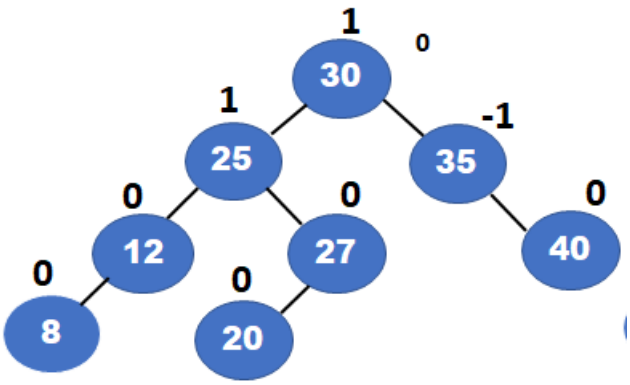
1. AVL trees
2. B-tree
3. Red Black trees

All these data structures force the tree to remain balanced and therefore guarantee performance.

AVL Tree

An AVL tree is height balanced Binary search tree invented by Adelson-Velskii and Landis. In AVL tree the heights of left and right sub-trees of a root differ by at-most one. Every node in the AVL tree is associated with balance factor that is left higher, equal-height or right-higher accordingly ,respectively as the left subtree has height greater than, equal to, or less than that of the right subtree.

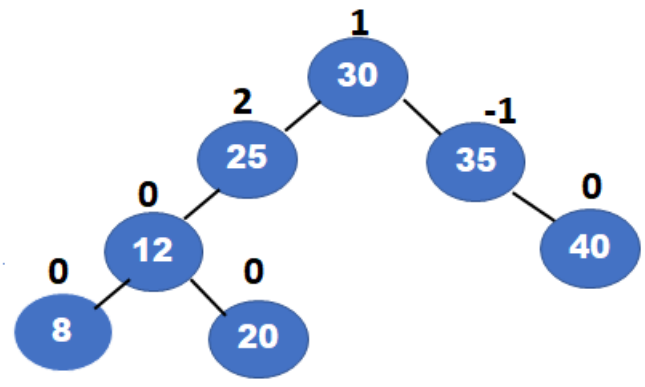
Example of AVL tree:



AVL tree

The tree shown above is an AVL tree as the difference between height of left subtree and right subtree of every node is at most one.

Example for not an AVL tree



Not an AVL tree

The above figure is not an AVL tree as the difference between height of left subtree and right subtree of root node is 2

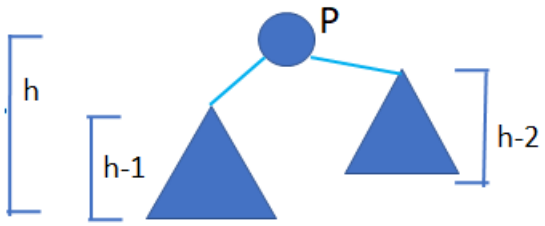
An AVL tree has the following properties:

1. The sub-trees of every node differ in height by at most one.
2. Every sub-tree is an AVL tree.
3. Height of AVL tree is always logarthmic in n.

An AVL tree has balance factor calculated at every node. For every node, height of left and right sub-trees can differ by no more than 1.

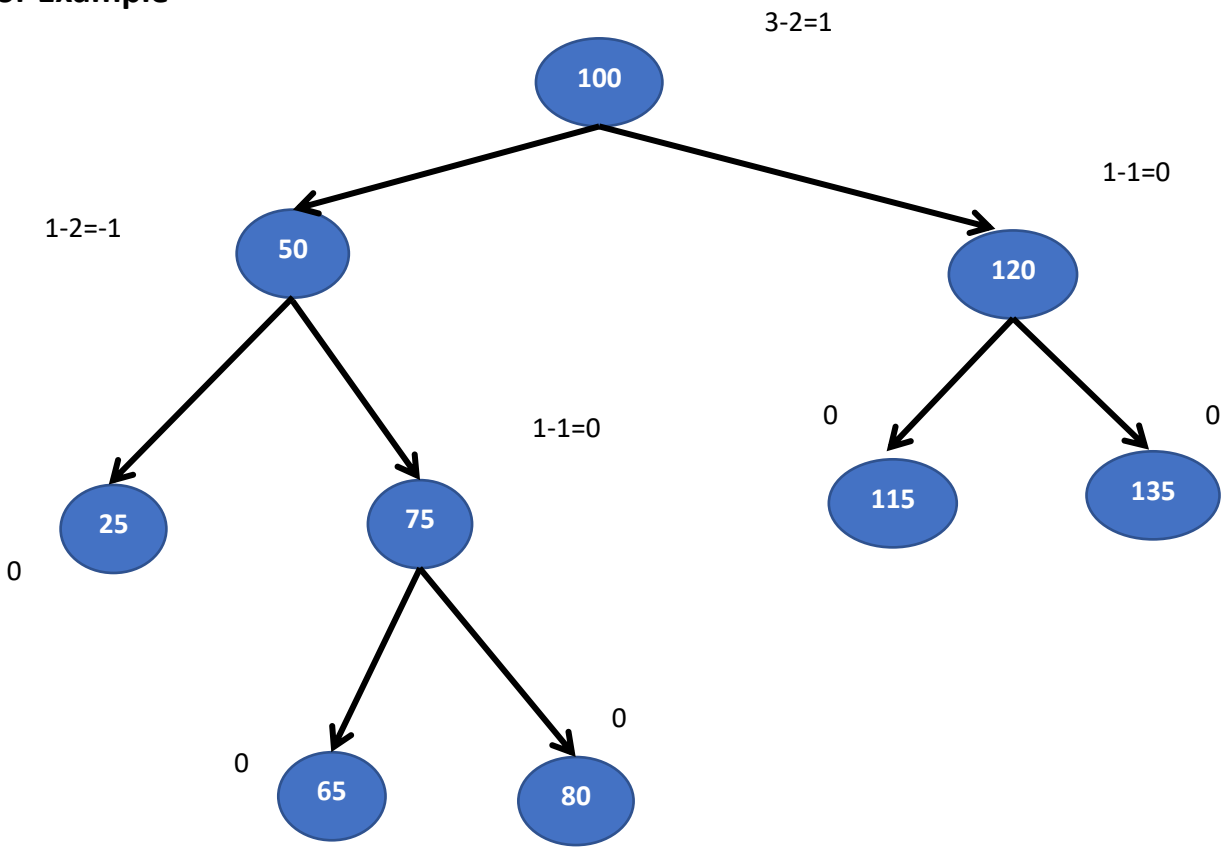
The balance factor of any node in an AVL tree is given by:

Balance factor=Height (left subtree)-Height (right subtree)



Balance factor = (h-1)-(h-2)

For Example



AVL tree rotations:

To balance the AVL tree after the insertions and deletion operations the four kinds of rotations are used. Rotations are an adjustment to the tree, around a node, that maintains the required ordering in the binary search tree.

1. Single Rotation
   a) Left rotation
   b) Right rotation
2. Double Rotation
   a) Left-Right rotation
   b) Right –Left rotation

1. Left Rotation

The tree is rotated left side to rebalance the tree, when a node is inserted into the right subtree of the right subtree. For example if the insertion order is 3, 5 and 7 the tree becomes imbalanced after the insertion of node 7. So we perform Left rotation (rotate in anti- clockwise) to balance the tree.

For example

Right unbalanced tree Left Rotation Balanced tree

2. Right Rotation

The tree is rotated right side to rebalance the tree, when a node is inserted in the left subtree of the left subtree. For Example if the insertion order is 8,6,4 the tree becomes imbalanced after the insertion of node 4 so to balance the tree we perform Right rotation(rotate clockwise).

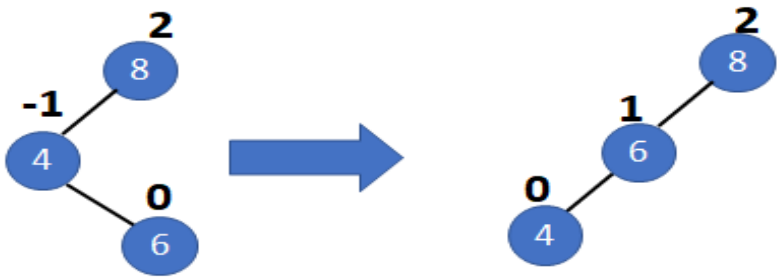
Left unbalanced tree right rotation Balanced tree

3. Left-Right Rotation

In Left-Right Rotation two rotations are performed to balance the tree. The tree is first rotated on left side and then on right side.

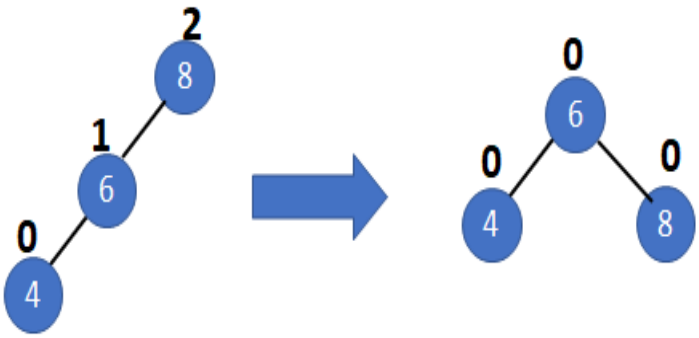
For example in the below unbalanced tree 8 is the root node, 4 is the left child of 8 and 6 is the Right child of 4. After left rotation on 4 and 6 ,the tree

structure looks like as shown below 8 is the root node,6 is the left child of 8 and 4 is the left child of 6



Unbalanced tree Left rotation unbalanced tree

The tree obtained after left rotation is unbalanced so again the tree is rotated right side inorder to balance the AVL tree. So a balanced tree with 6 as the root node,4 will become the left child of 6 and 8 is the right child of 8 is obtained.

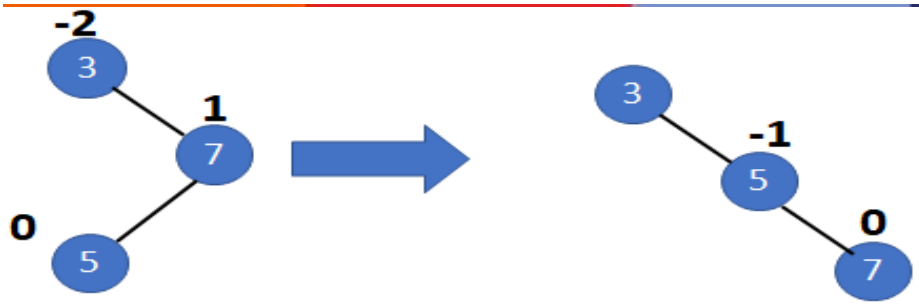


Unbalanced tree Right rotation Balanced Tree

4. Right-Left Rotation

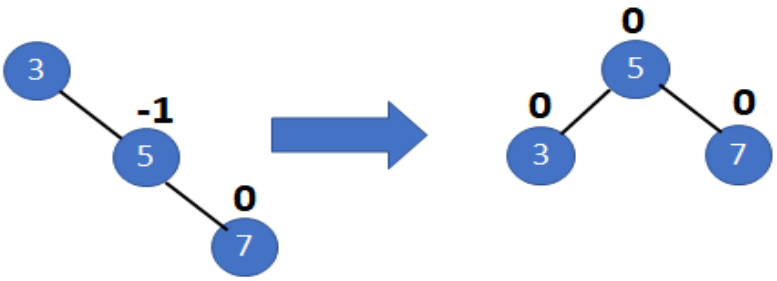
In Right-Left Rotation, first the tree is rotated right side and then to the left side.

For Example in the below unbalanced graph, 3 is the root node 7 is the right child of 3 and 5 is the left child of 7.After applying first right rotation on 7 and 5, then we get a unbalanced tree with 3 as the root node, 5 is the right child of 3 and 7 is the right child of 5.



Unbalanced tree Right rotation unbalanced tree

Then the tree is rotated left side inorder to obtain the balanced tree where 5 is the root node, 3 will be left child of 5 and 7 will be right child of 5.



Unbalanced tree Left rotation Balanced tree