AVL Tree

AVL tree is a self-balancing binary search tree in which each node maintains extra information called a balance factor whose value is either -1, 0 or +1.

AVL tree got its name after its inventor Georgy Adelson-Velsky and Landis.

**Balance Factor**

Balance factor of a node in an AVL tree is the difference between the height of the left subtree and that of the right subtree of that node.

Balance Factor = (Height of Left Subtree - Height of Right Subtree) or (Height of Right Subtree - Height of Left Subtree)

The self balancing property of an avl tree is maintained by the balance factor. The value of balance factor should always be -1, 0 or +1.

An example of a balanced avl tree is:

|  |
| --- |
| avl tree |
| Avl tree |

**Operations on an AVL tree**

Various operations that can be performed on an AVL tree are:

**Rotating the subtrees in an AVL Tree**

In rotation operation, the positions of the nodes of a subtree are interchanged.

There are two types of rotations:

**Left Rotate**

In left-rotation, the arrangement of the nodes on the right is transformed into the arrangements on the left node.

Algorithm

1. Let the initial tree be:

|  |
| --- |
| left-rotate |
| Left rotate |

1. If y has a left subtree, assign x as the parent of the left subtree of y.
2. If the parent of x is NULL, make y as the root of the tree.
3. Else if x is the left child of p, make y as the left child of p.
4. Else assign y as the right child of p.

|  |
| --- |
| left-rotate |
| Change the parent of x to that of y |

1. Make y as the parent of x

|  |
| --- |
| left-rotate |
| Assign y as the parent of x. |

**Right Rotate**

In left-rotation, the arrangement of the nodes on the left is transformed into the arrangements on the right node.

1. Let the initial tree be:

|  |
| --- |
| right-rotate |
| Initial tree |

2.If x has a right subtree, assign y as the parent of the right subtree of x

|  |
| --- |
| right-rotate |
| Assign y as the parent of the right subtree of x |

1. If the parent of y is NULL, make x as the root of the tree.
2. Else if y is the right child of its parent p, make x as the right child of p.
3. Else assign x as the left child of p.

|  |
| --- |
| right-rotate |
| Assign the parent of y as the parent of x. |

1. Make x as the parent of y.

|  |
| --- |
| right-rotate |
| Assign x as the parent of y |

**Left-Right and Right-Left Rotate**

In left-right rotation, the arrangements are first shifted to the left and then to the right.

1. Do left rotation on x-y

|  |
| --- |
|  |
|  |