

# ADS Lab - 4

ALL → tree Insertion & Deletion

~~node~~

class Treenode (object):

def \_\_init\_\_ (self, key):

self.key = key

self.left = ~~0~~ None

self.right = None

self.height = 1

class AVL (object): → AVL Tree Class

def insert\_node (self, root, key) → Insert Method

if not root:

return Treenode (key)

elif key < root.val:

root.left = self.insert (root.left, key)

else:

root.right = self.insert (<sup>root.right</sup> root.right, key)

root.height = 1 + max (self.getheight (root.left),  
<sup>self</sup> self.getheight (root.right))

balance = get balance (root)

if (balance > 1 and key > root.left.val):

root.left = self.Left Rotate (root.left)

return self Left Rotate (root)



```

if (balance < -1 and key < root.right.val):
    root.right = self.rightRotate(root)

if (balance > 1 and key < root.left.val):
    return self.rightRotate(root)

if (balance < -1 and key > root.left.val):
    return self.leftRotate(root)

return root

```

```

def rightRotate(self, rootz): → For Right Rotation
    y = root.right
    T2 = y.left

    y.left = z
    root.right = T2

    rootz.height = 1 + max(rootgetHeight(z.left),
                        rootgetHeight(root.right))

    y.height = 1 + max(rootgetHeight(y.left),
                      rootgetHeight(y.right))

    return y

```



~~def rightRotate (self, root):~~

def leftRotate (self, z):  $\rightarrow$  For Left Rotation

y = z.right  
T2 = y.left

y.left = z  
z.right = T2

z.height = 1 + max (getHeight(z.left),  
getHeight(z.right))

y.height = 1 + max (getHeight(y.left),  
getHeight(y.right))

return y

def getHeight (self, root):  $\rightarrow$  Returns the Height  
if not root:

return 0

return root.height

def getBalance (self, root):  $\rightarrow$  Returns Balance Factor  
if not root:

return 0

return self.getHeight(root.left)

- self.getHeight(root.right)



def

def delete(self, root, key):

if not root:  
return root

elif key > root.val:  
root.left = delete(root.left, key)

elif key < root.val:  
root.right = delete(root.right, key)

else

if root.left == None:  
temp = root.right  
root = None  
return temp

elif root.right == None:  
temp = root.left  
root = None  
return temp

if root.height == 1 + max(getHeight(root.left),  
getHeight(root.right))

balance = getBalance(root)

if balance > 1 and getBalance(root.right) < 0:  
return leftRotate(root)

if balance > 1 and getBalance (root.left) >= 0:  
return rightRotate (root)

if balance > 1 and getBalance (root.left) < 0:  
root.left = leftRotate (root.left)  
return rightRotate (root)

if balance < -1 and getBalance (root.right) > 0:  
root.right = rightRotate (root.right)  
return leftRotate (root)

return root.