AVL TREES

#FLOW OF THE CODE FOR AVL TREES

STEP 1: INSERT NODE AT APPROPRIATE LOCATION.

STEP 2 : CALCULATE HEIGHT OF EACH NODE.

STEP 3 : CALCULATE BALANCED FACTOR.

STEP 4 : IF REQUIRED PERFORM ROTATION.

Code:

#AVL tree

class node:

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

self.value=data

self.left=None

self.right=None

self.height=1

def In\_order(root):

if not root:

return

In\_order(root.left)

print(root.value,end=' ')

In\_order(root.right)

def insert(root,Super):

if not root:

return node(Super)

if Super<root.value:

root.left=insert(root.left,Super)

else:

root.right=insert(root.right,Super)

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

BF=getBF(root)

#RR

if BF>1 and Super<root.left.value:

return rightRotate(root)

#LR

if BF>1 and Super>root.left.value:

root.left=leftRotate(root.left)

return rightRotate(root)

#LL

if BF<-1 and Super>root.right.value:

return leftRotate(root)

#RL

if BF<-1 and Super<root.right.value:

root.right=rightRotate(root.right)

return leftRotate(root)

return root

def ght(root):

if not root:

return 0

return root.height

def getBF(root):

if not root:

return 0

return ght(root.left)-ght(root.right)

def leftRotate(A):

B=A.right

temp=B.left

B.left=A

A.right=temp

A.height=1+max(ght(A.left),ght(A.right))

B.height=1+max(ght(B.left),ght(B.right))

return B

def rightRotate(A):

B=A.left

temp=B.right

B.right=A

A.left=temp

A.height=1+max(ght(A.left),ght(A.right))

B.height=1+max(ght(B.left),ght(B.right))

return B

if \_\_name\_\_=="\_\_main\_\_":

VL=[19,99,75,7,21,34,38,27,134,100,29,0,12,17,143]

root=None

for i in VL:

root=insert(root,i)

print('\nIn\_order traversal ')

In\_order(root)

Output:

In\_order traversal

0 7 12 17 19 21 27 29 34 38 75 99 100 134 143