LCS[i,j] = 1 + LCS[i-1,j-1]

else

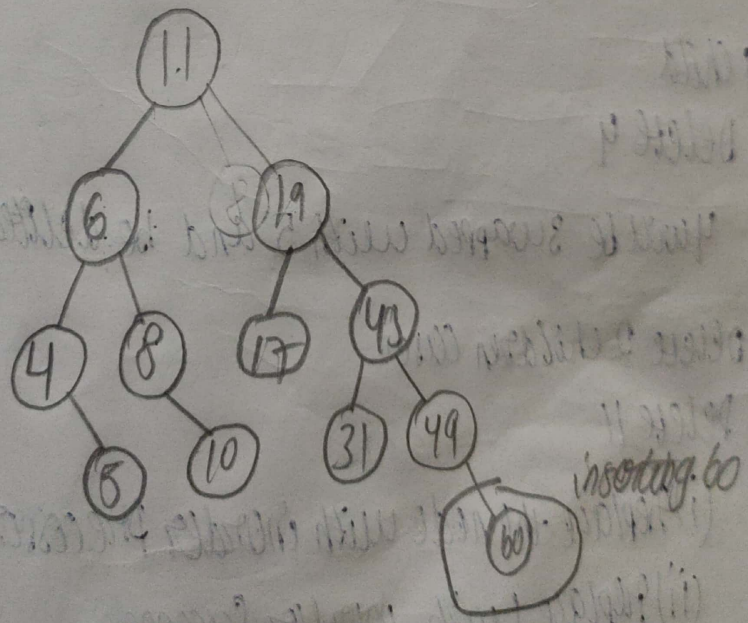
LCS[i,j] = max(LCS[i-1,j], LCS[i,j-1])

UNIT-3

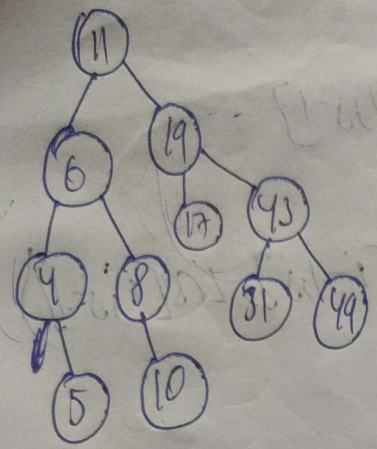
BINARY SEARCH TREE:

→ atmost 2 child nodes

11 6 8 19 4 10 5 17 43 49 31



DELETION:



Deletion:

(1) 0 child

(2) 1 child

(3) 2 child

(1) 0 child

Delete 31

(i) Simply delete the node

(2) 1 child

Delete 4

4 will be swapped with 5 and be deleted

(3) Delete 2 children case:

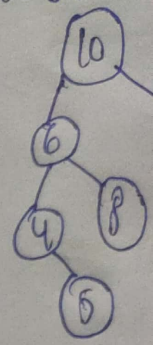
Delete 11

(i) Replace the node with inorder predecessor

(ii) Replace with inorder successor

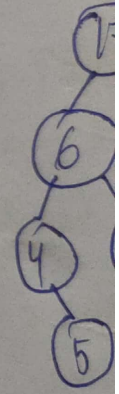
Inorder Predes

the largest el



Inorder Su

the smal



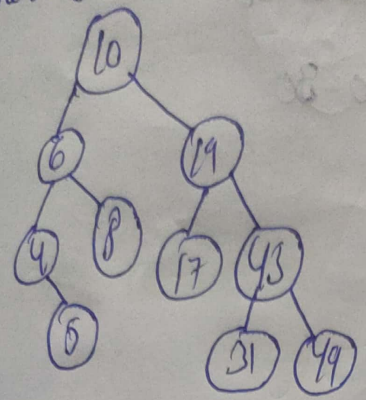
SEARCH:

(1) key

(2) ke

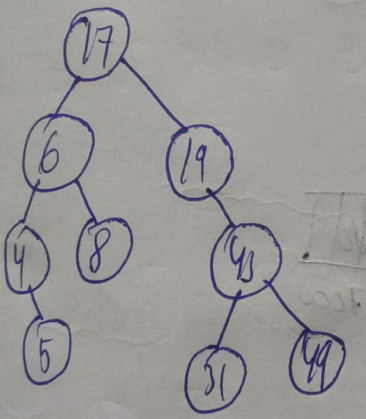
Inorder predecessor:

the largest element from left subtree



Inorder Successor:

the smallest element from right subtree



SEARCH:

① key >= Root

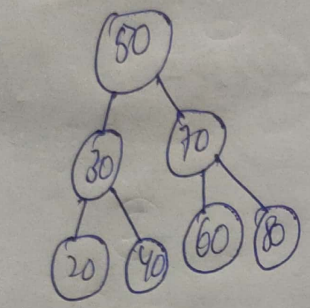
② key < Root

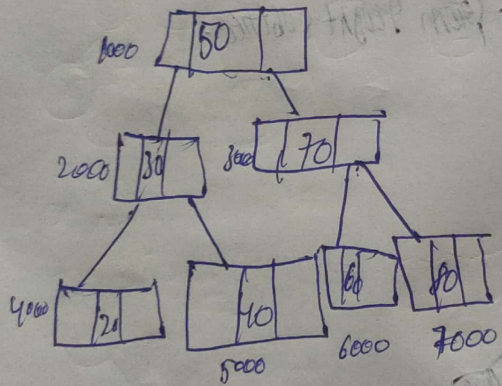
search space — left subtree

① key > Root

Search space - Right Subtree

50 30 70 20 40 60 80





INORDER TRAVERSAL:

```
inorder (root -> left)
display (root -> data)
inorder (root -> right
```

PREORDER:

```
display(root -
preorder (root
pre order (root
```

post order TRAVERS

```
Postorder (root
Postorder (roo
display (roo
```

AVL Trees:

1- It is a BST.

2. Height of le

= {-1, 0, 1}

3. No duplicate

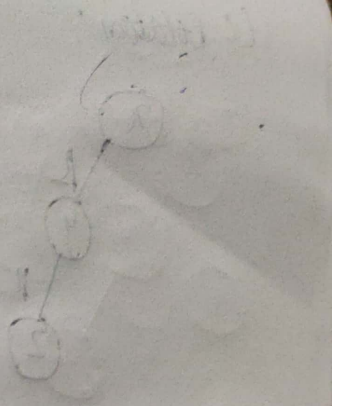
After insertion

PREORDER:

display(root -> data)

preorder(root -> left)

preorder(root -> right)

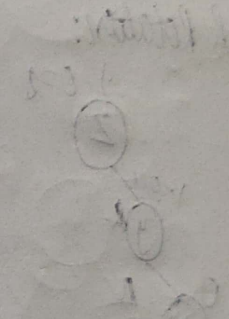


POSTORDER TRAVERSAL

postorder(root -> left)

postorder(root -> right

display(root -> data)



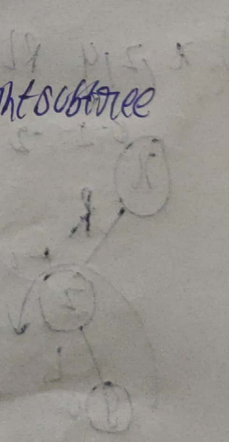
AVL TREES:

1- It is a BST

2. height of left subtree - height of rightsubtree

= {-1, 0, 1} for each node

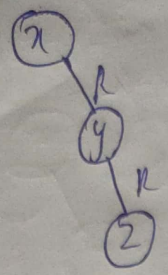
↳ balance factor

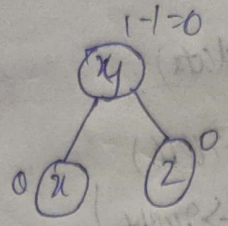


3. No duplicate elements

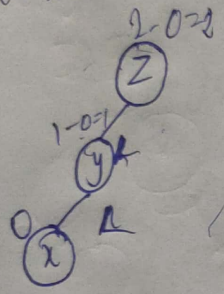
After insertion, balance factor is to be found

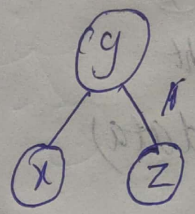
LL Rotation :



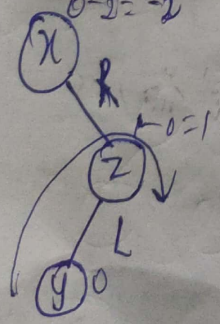


RR Rotation :

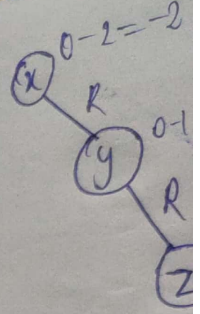




(3) x, z, y RL Rotation

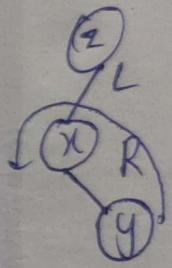


Right Rotation for z and y



(4) LR rotat

z, x, y

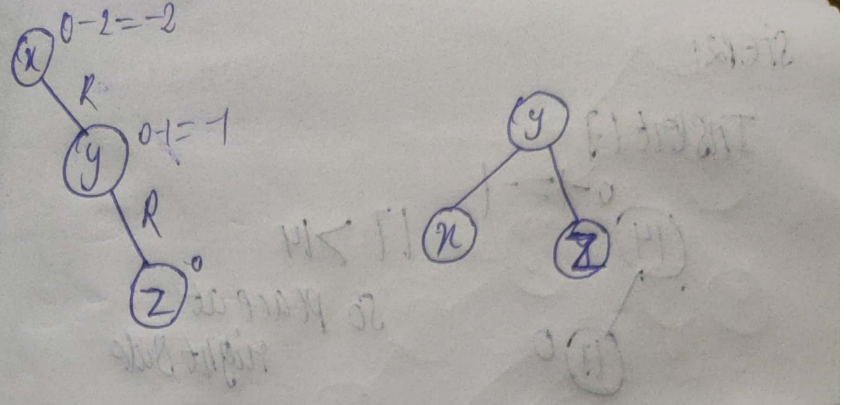


AVL Inse

14, 17, 1

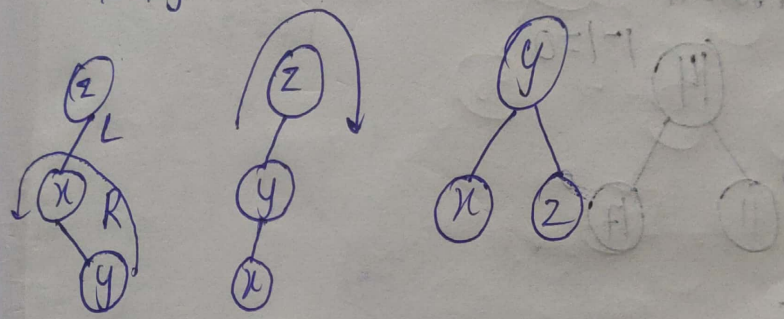
STEP 1:

Insert



(4) LR rotation:

z, x, y



AVL INSERTION

14, 17, 11, 7, 53, 4, 13, 12, 8, 60, 19, 16, 20

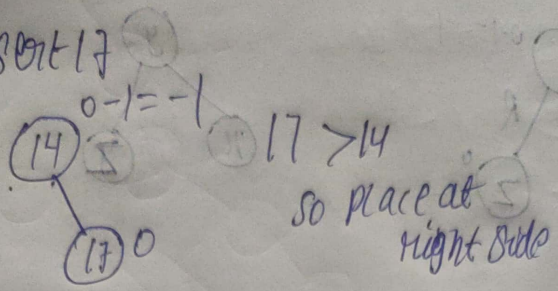
STEP1:

Insert 14

(14) → balance factor = 0-0=0

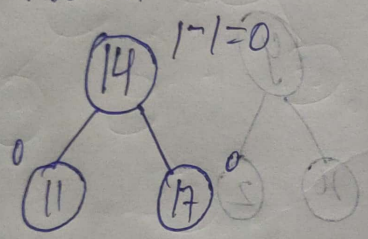
STEP2:

Insert 17



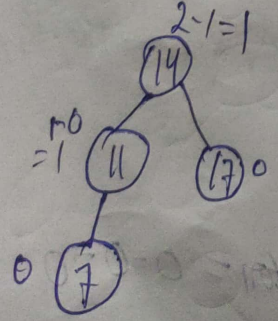
STEP3:

Insert 11

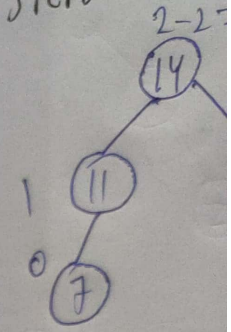


STEP4:

Insert 7

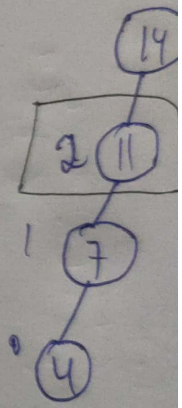


STEP5:



STEP6:

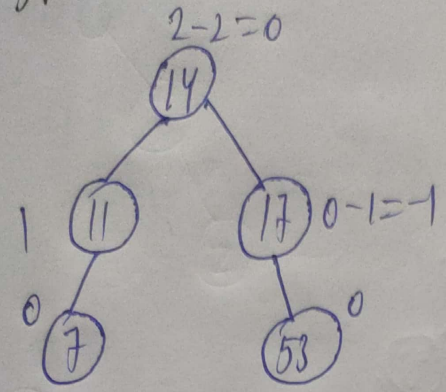
Insert 4



node 11 is

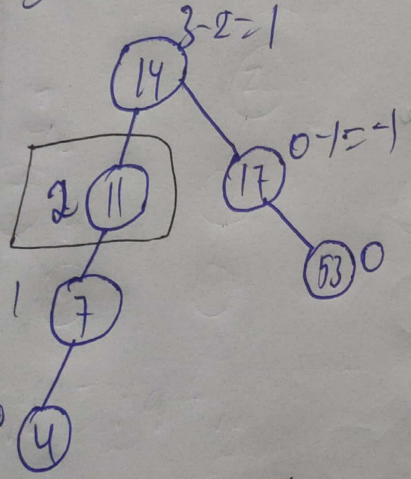
As it is

STEP5:



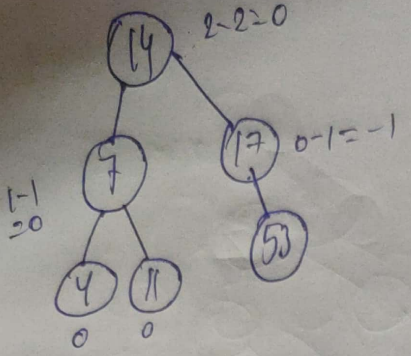
STEP6:

Insert 4

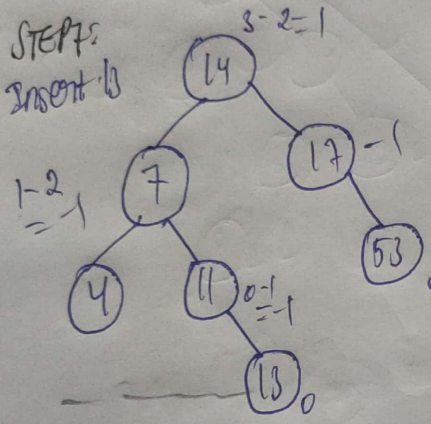


node 11 is Critical

As it is LL form then right rotation is then

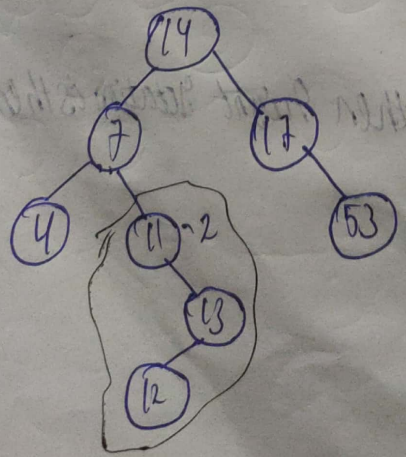


STEP7:
Insert 13

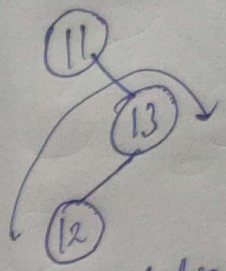


STEP8:

Insert 12

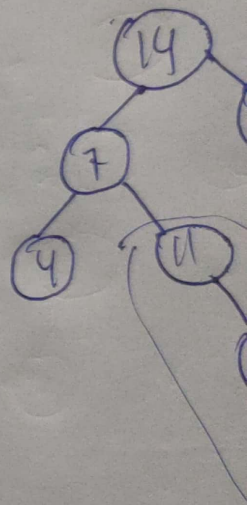


Node 14 critical node

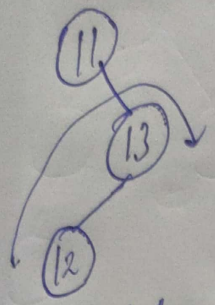


RL rotation

(i) right rotate

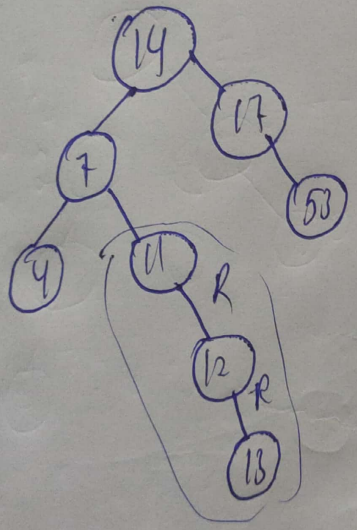


Node as critical node



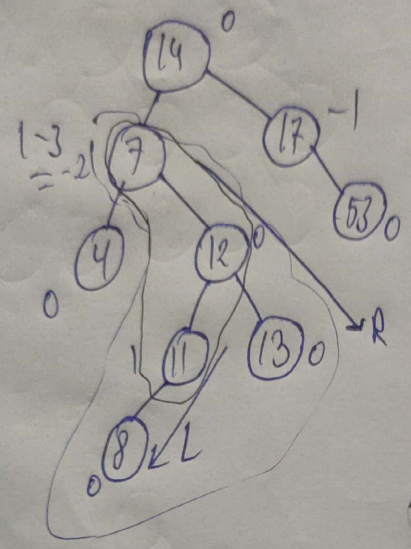
RL rotation

(i) right rotation

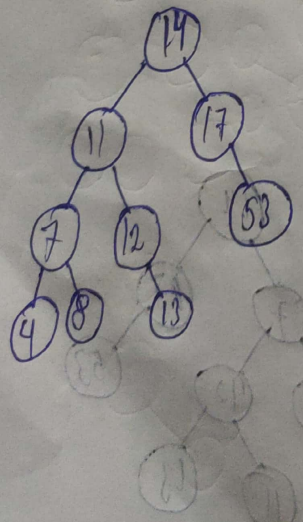


STEP 9:

Insert 8

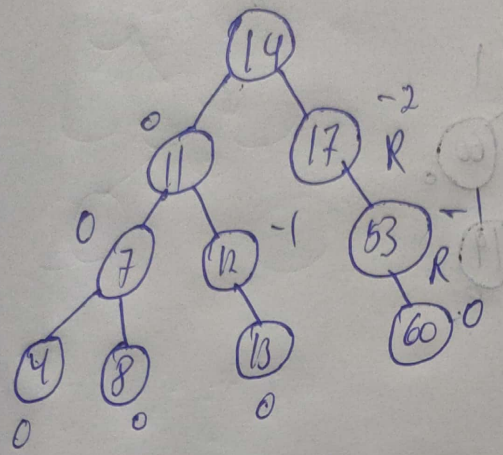


7, 11, 12 → median is 11



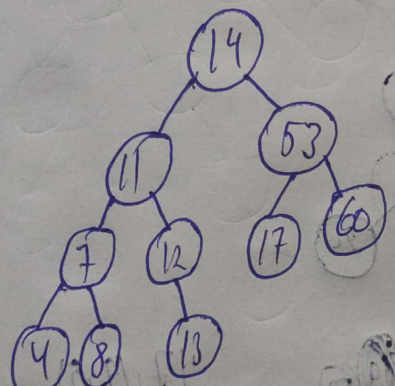
STEP 01:

Insert 60



RR Rotation occurs

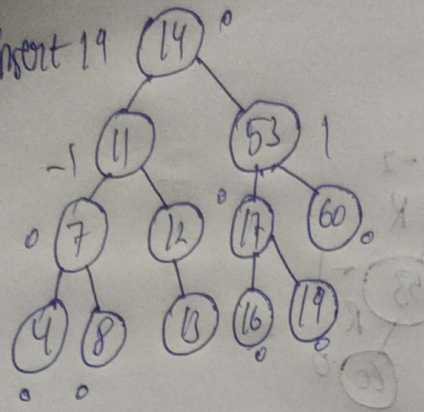
17 / 53 / 60 → 53 middle

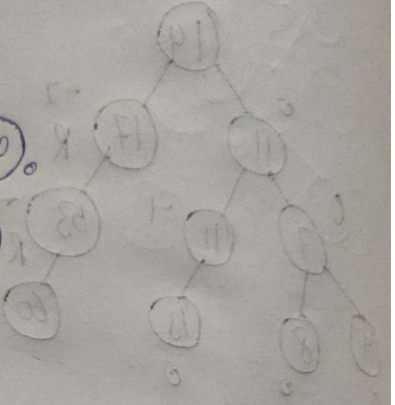


Scanned with CamScanner

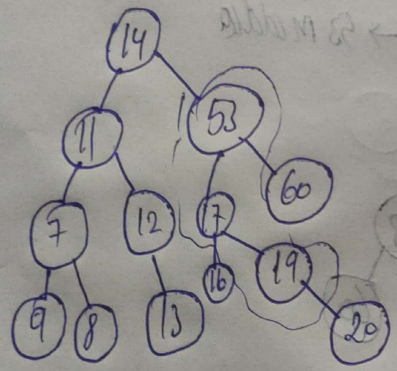
STEP II:

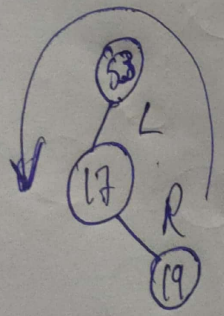
Insert 19

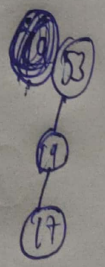


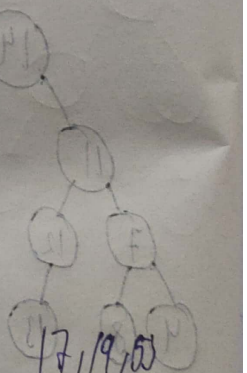


2. Insert 20

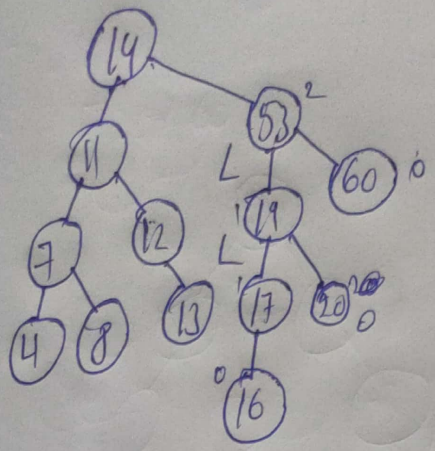




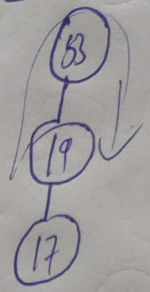


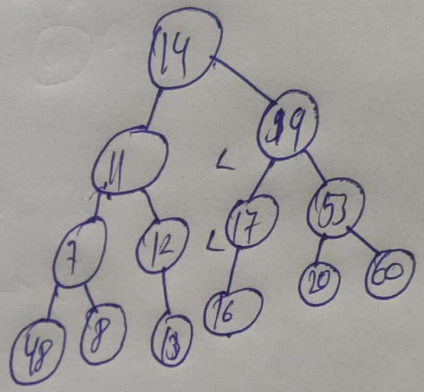


17/19/53
median
19



LL





AVL DELETION: