# AVL TREE ALGORITHMS

into height balanced BST. Suppose initially there is a height balanced Binary Tree.

Hhenever a mode is inserted into it (or deleted from it), it may become unbalanced.

## Following Steps need to be adopted.



- 1. Insert mode into Binary Search Tree: >
- 2. Compute the Balance factor: +On the path starting from the goot node to the mode newly inserted, compute the Balanced factors of each mode. It can be verified that a change in Balance Factors will occur only in this path.
- 3. Decide the pivot node: + on the path as traced in Step2, determine whether the absolute value of any mode's Balance Factor is switched from 1 to 2. If so, the tree becomes Unbarractor. The mode which has its absolute value of Balance Factor switched from 1 to 2.

  marked 3s a Special Node and called the Pivot Node.

There may be more than one node which has its Balance Factor, I bil switched from 1 to 2, but the nearest yode to the newly Prisoned node will be the pivot yode.

4. Balance the Unbalance Toree: -> It is necessary to manipulate pointers centred at the pivot mode to bring the tree back into height balance. This pointer manipulation is well known as AVL Rotation.

where don't will be discussioned where

the second of th

gold ateriance of the said along the said the state

The fact the basis and in the salar days of the salar

Elighten with the place representation and with manifestion

epresentation god at books of an alleganth on a taken dowing with all a

tertained as a stail a content cretain part to environ details will restlegist

growth hardwise retail and extent arrived

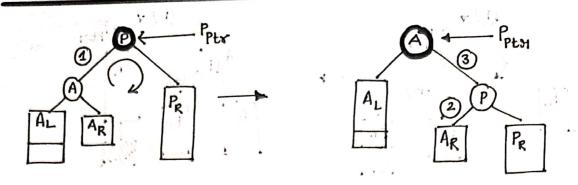
and are I war to me to give the band the bang of the

the second of the shape demands and home well to prove that have been

I should a method the same states of the soft of the soft and the same same

structured in and agreement to secretary and and and

### ALGORITHM LEFT TO LEFT ROTATION



20 at the Thirty W duth ! : 19 4 . . .

Augustina wight her our fix

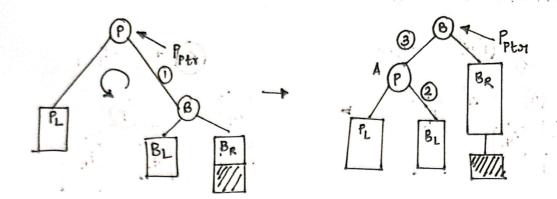
this et by the state of

INPUT: Pointer Pptr to the pivot node

OUTPUT: AVL Rotation corresponding to the unbalance due to insertion in the left sub-tree of the left child of Pptr.

Steps:

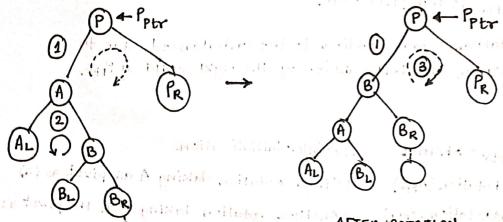
#### ALGORITHM RIGHT TO RIGHT ROTATION



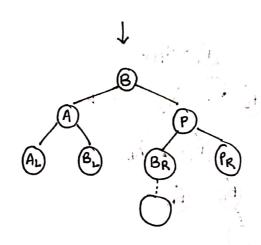
INPUT: Pointer Ppty to the Pivot Node

OUTPUT: AVL Motation corresponding to the unbalanced due to insertion at the Hight sub-tree of the right child of Petr.

Steps:



AFTER ROTATION RIGHT- TO-RIGHT



AFTER ROTATION LEFT- To-LEFT

INPUT: Pointer Pptr to the pivot node.

OUTPUT: AVL Rotation corresponding to the unbalance due to Insertion in the right sub-tree of the left child of Pptr.

#### Steps:

1. Aptm = Pptr -> LCHILD

11 Pointer Initialization 28 (1)

2. Right To Right Rotation (Aptr)

11 Single rotation taking A as pivot as (2)

3. Left to Left Rolation (Pptr)

11 Another votation taking P as pivot as (3)

4. Stop

#### 4. ALGORITHM RIGHT TO LEFT ROTATION

INPUT: Pointer Petr to the pivot node.

OUTPUT: AVL Rotations corresponding to the unbalanced due to insertion in the left subtree of the right child of Petr.

Steps:

1. Aptr = Pptr → RCHILD

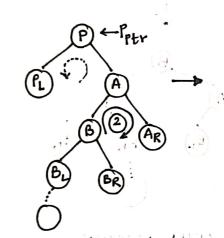
11 Pointer mihalization

2. Left To Left Rotation (Aptr) /1 Single Rotation taking A as pivot as (2)

3. Right To Right Rotation (Pptr) 1/ Another rotation taking P as the pivot as (3)

s, Aleachdine day leadang saraten

4. Stop

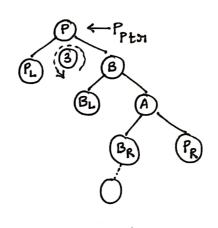


in said tot and anothering and of

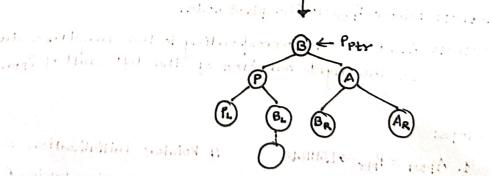
registe their that with

A tacket in the Anthony in

AFTER INSERTION



AFTER ROTATION LEFT- TO-LEFT



and the single military shall it APTER ROTATION THE TENER . RIGHT-TO- RIGHT

e. Leather at Epidles (1910)

```
Steps:
 1. ptr=Root
2. If (ptr= NULL) then
        ptr = Nptr
3.
        ptr -> HEIGHT=1
4.
        Return ()
5.
    Else
6.
       If (N_{Ptr} \rightarrow DATA < ptr \rightarrow DATA) then
I.
           InsertHBT (ptr > LCHILD, Nptr)
8.
           Lptr = ptr > LCHILD
9.
           Rptr = ptr > RCHILD
10.
           If (Rpt = NULL) then
11.
                he=0
12.
13.
            Else
                hR = Rptr >HEIGHT
14.
                h_= Lptr > HEIGHT
15.
                 bf = hL-ha
16.
                1f (bf=2) then
17.
                    If (Nptr + DATA < Lptr > DATA) then
18.
                       Left to left lotation (ptr)
19.
                    Else
20.
                        Letto Right Rotation (ptr)
21.
                   Endsf
22.
                    Ptot -> HEIGHT = ComputeHeight(ptr)
23.
24.
                  Endif
```

Endif

25.

ALLYS ...

Athur.

```
26. Else
         If (Npty + DATA > ptr -> DATA) then
                                                          Bilingad tuntungsa
27.
            Insert HBT (ptr > RCHILD, Nptr)
28.
            Rptr = ptr → RCHILD
29.
30.
            Lptr = ptr > LCHILD
 31.
            If ( Lpty = NULL) then
                                                           t meane was
 32.
               h_ = 0
 33.
            Elsc
 34.
                h_= Lptr > HEIGHT
 35.
                hR = Rptr > HEIGHT
                bf= h_-hR
 36.
 37.
                If (bf = -2) then
                                                                              . . . .
 38.
                  If (Nptr -> DATA > Rptr -> DATA) then
                                                                              . 3 !
                     RIGHT TO RIGHT ROTATION (PEY)
 39.
                                                                              . 1 .
40.
                  Else
                                                                              . 02
                     RIGHT TO LEFT ROTATION (ptr)
41.
                                                    padaga - 23
42.
                  Endly
43.
                   ptr > HEIGHT = computeHeight (ptr)
                        ing (what sady i was to sady) for
                                                                              . * 5
44.
              Enalf
                                  Lott by bathle taller, (pla)
 45.
             ENdif
                                                                              * 1 A
 46.
          Else
              print Npt -> DATA" is already exist in the tree"
 47.
                                                                              * 1 C
  40.
        Endit
                                                         Herit
       Endly
  49.
                          (, etg) ry miningania w Mristle - 114
  50. Endif
```

51. Stop.

Maril