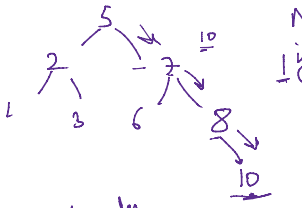
30 Jan

30 January 2022 19:54

Binary Search Tree :-

=> max(root.left) < root
=> min(root.right) > root

data = 10

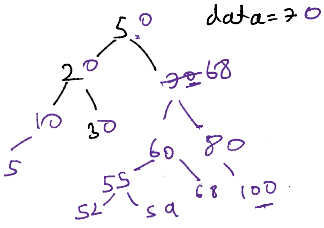


n = no. of nodes

$2^3 = 8$ | $\log(8) = 3$

T.C => height of the tree.

```
Node addBST(root, data) {
1 if(root == null) return new Node(data);
2 if(root.val > data) {
    root.left = addBST(root.left, data);
3 } else if(root.val < data) {
    root.right = addBST(root.right, data);
  }
return root;
}
```



=> Find the node
=> Find the replacement

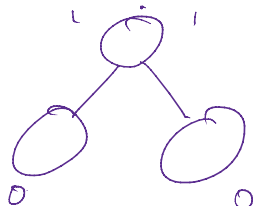
a) if removenode is a leaf.
b) if " has only one child
c) " " " both child.

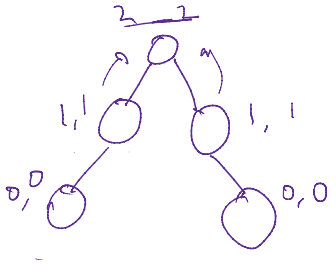
```
Node removalBST(n, data) {
- if(n.val > data) {
n.left = removalBST(n.left, data);
- } else if(n.val < data) {
n.right = removalBST(n.right, data);
} else {
if(isLeaf(n)) {
return null;
} else if(n.left != null && n.right == null) {
return n.left;
} else if(n.right != null && n.left == null) {
return n.right;
} else {
int max = getMax(n.left);
n.val = max;
n.left = removalBST(n.left, max);
}
return n;
}
```

Balanced Tree :-

bal. Factor = l.h − r.h

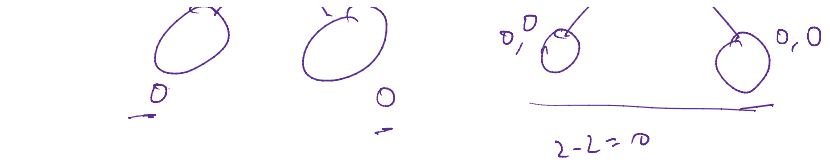
$-1 \le bf \le 1$





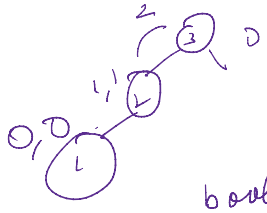
$-1 \le bf \le 1$

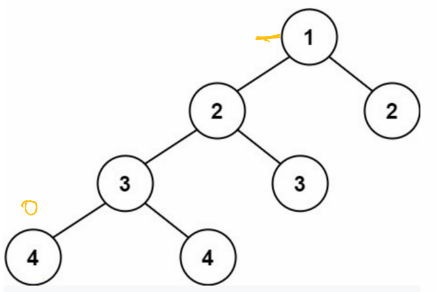
$-1, 0, 1$



$1 - 1 = 0$

$2 - 2 = 0$



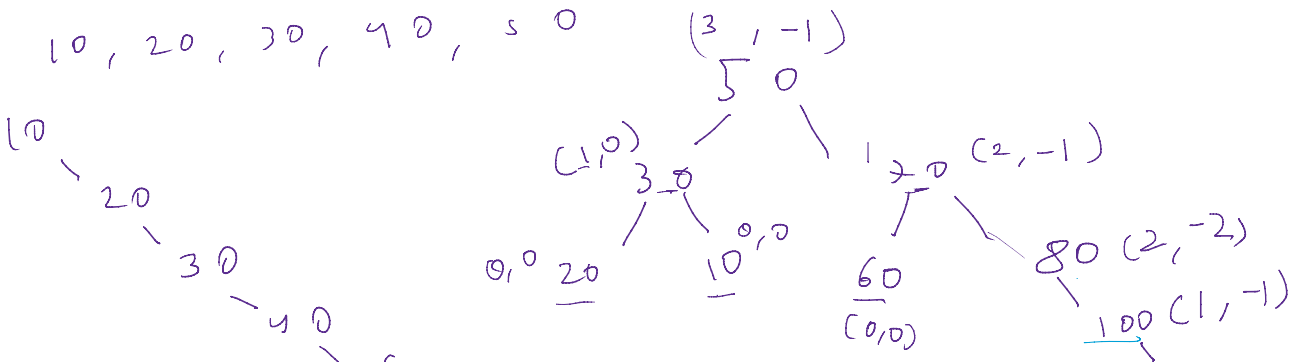


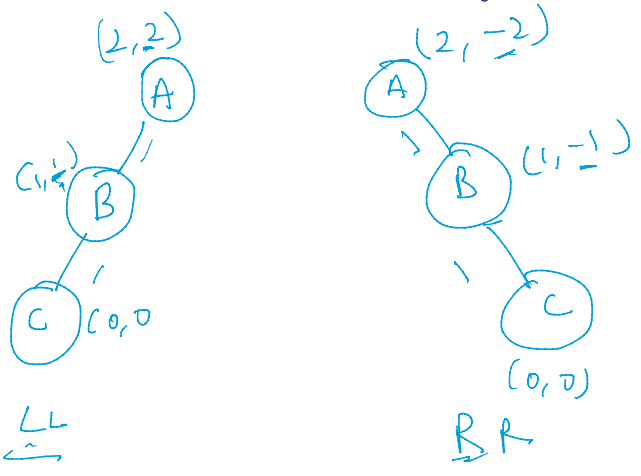

```
boolean isbalanced = true;

int bal(root) {
    if(root == null) return -1;
    l = bal(root.left);
    r = bal(root.right);
    bf = Math.abs(l - r);
    if(bf > 1) isbalanced = true;
    return max(l,h) + 1;
}
```

AVL Tree (Self-Balancing tree):

10, 20, 30, 40, 50

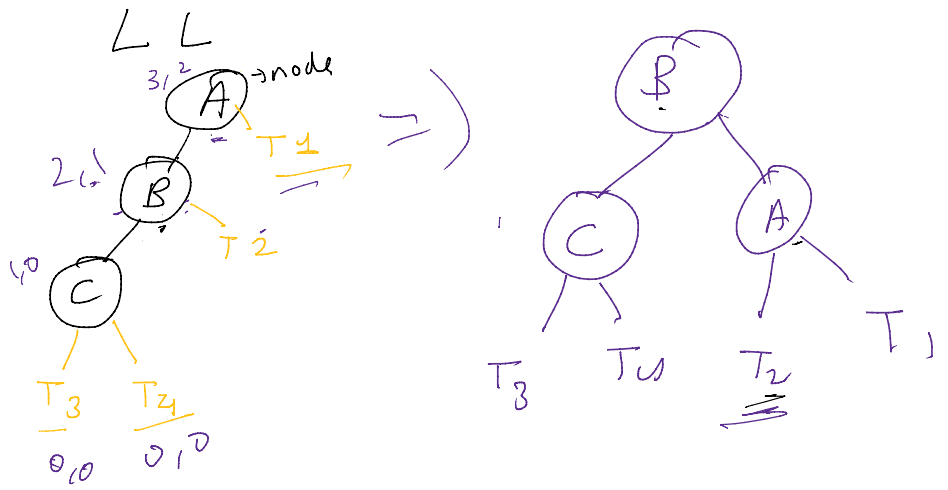




LL

2,2 → node

note LL (not A



node LL (node

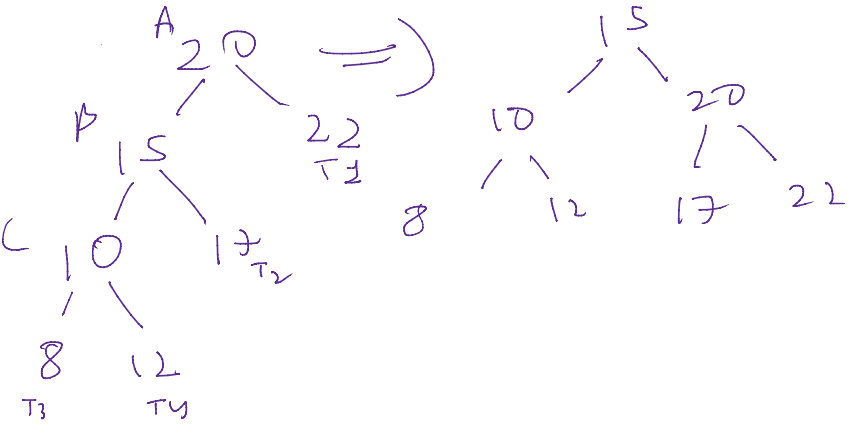
B = node.

node.left = B

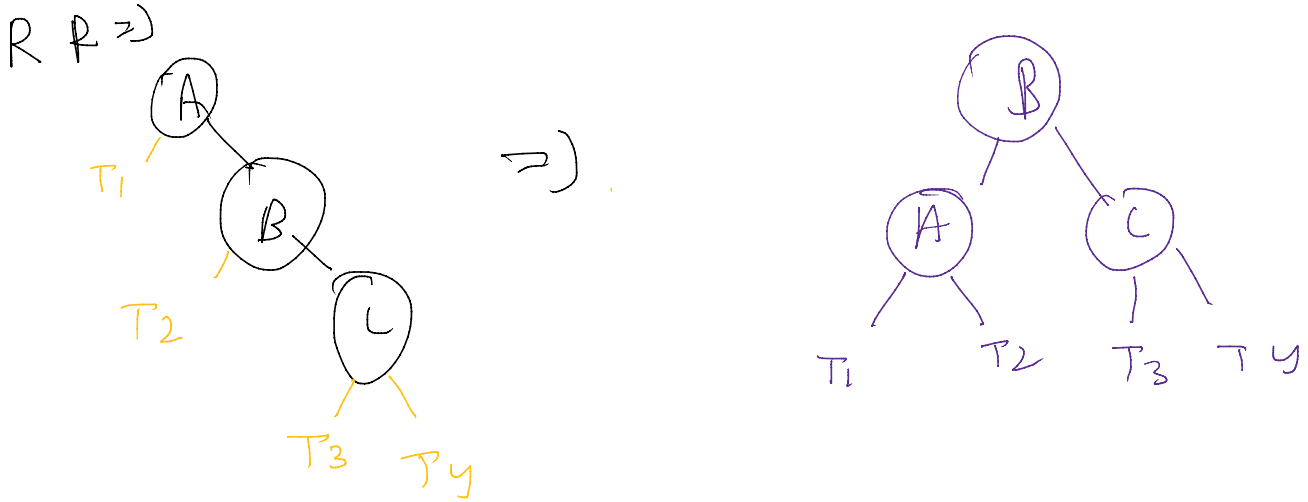
B.right =

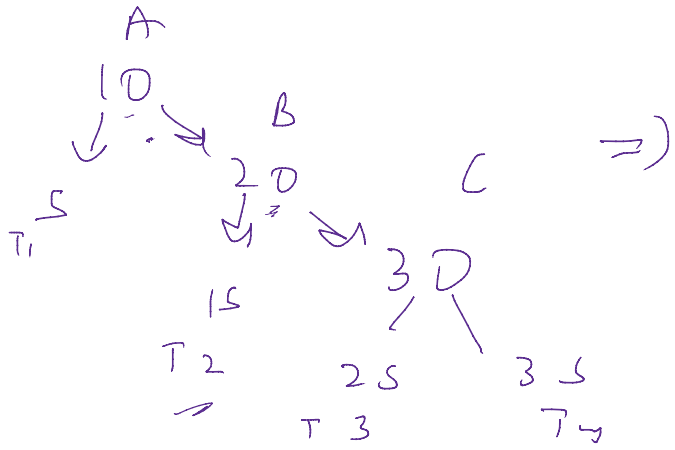
return B;

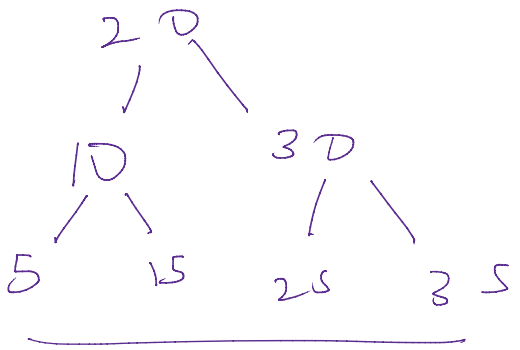
}



R R =>







left

right

node

```
node RR(node) {
  B = node.right
  node.right = B.left;
  B.left = node
    return B;
}
```