Today's Content:

- BST Basics
- BST only unique cases:
  - Search(k)
  - Insert()
  - Delete()
- isBST()
- isBalancedBST()
- TrimBST() : {If Time permits}

Tree construction discussed later

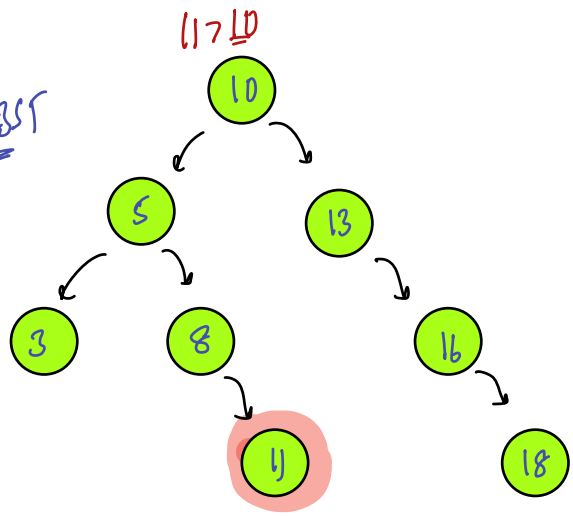
BST: Binary Search Tree

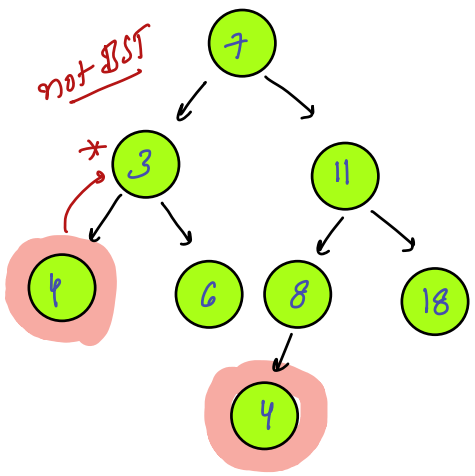
A Binary Tree is BST if

For all Nodes: All elements < node < All elements in LST in RST

Ex1:

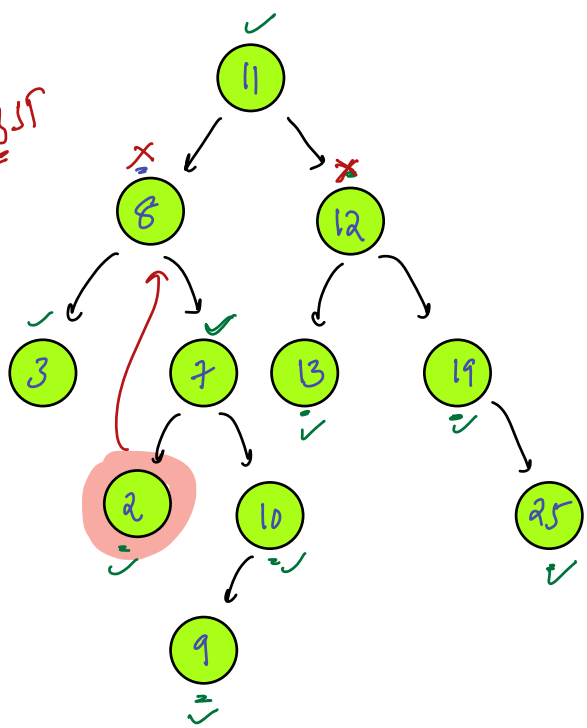
not BST

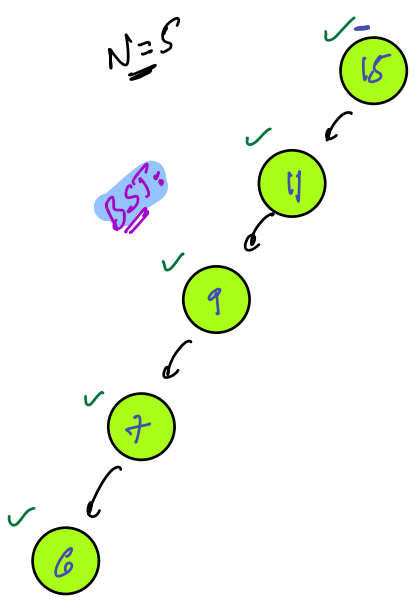




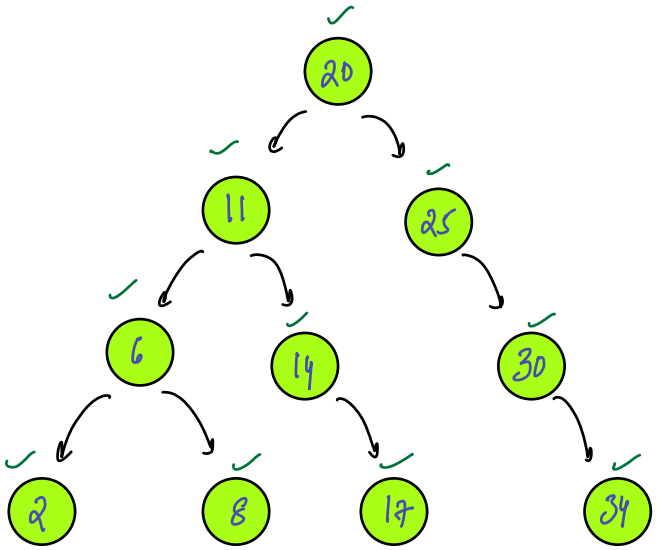
Ex3:

Not BST

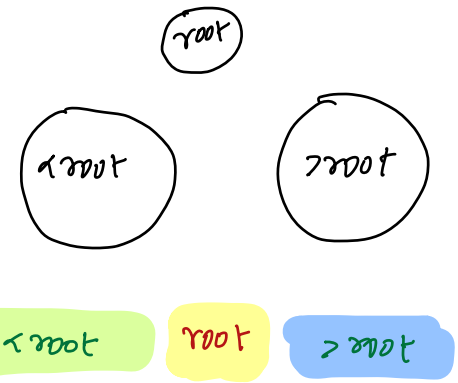




Interesting property:



Inorder : LDR



inorder: 2 6 8 11 14 17 20 25 30 34

obs: → Inorder in BST is sorted

↳ above applies only for BST

// Note: Do duplicates in Data

↳ With duplicates at last : 5mins

BST:

k = 9

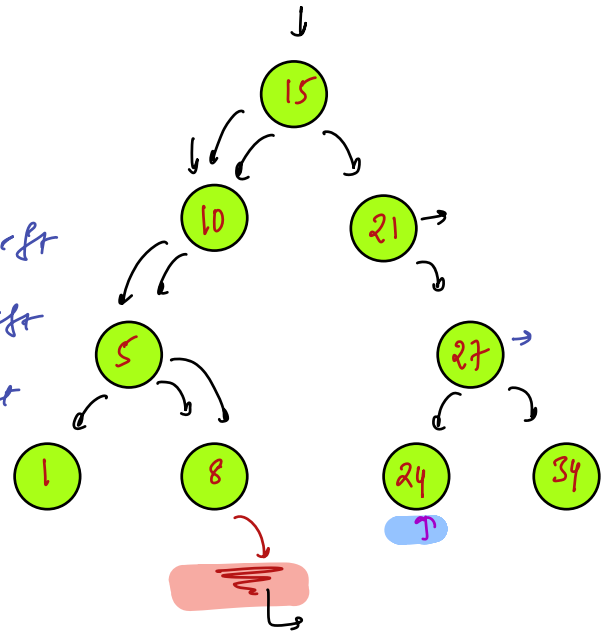
9 < 15 : goto left

9 < 10 : goto left

9 > 5 : goto right

9 < 8 : goto right

NULL :



k = 24

24 > 15 : (goto right)

24 > 21 : goto right

24 < 27 : goto left

```
bool Search (Node root, k) {                    → TODO Recursion

    while ( root != NULL) {

        if (root.data == k) { return True }

        if (root.data < k) {
            root = root.right
        }
        else if (root.data > k) {
            root = root.left
        }
    }
    return false
}
```

Height of Tree : H

TC: O(H)

Worst: H = (N-1) → O(N)

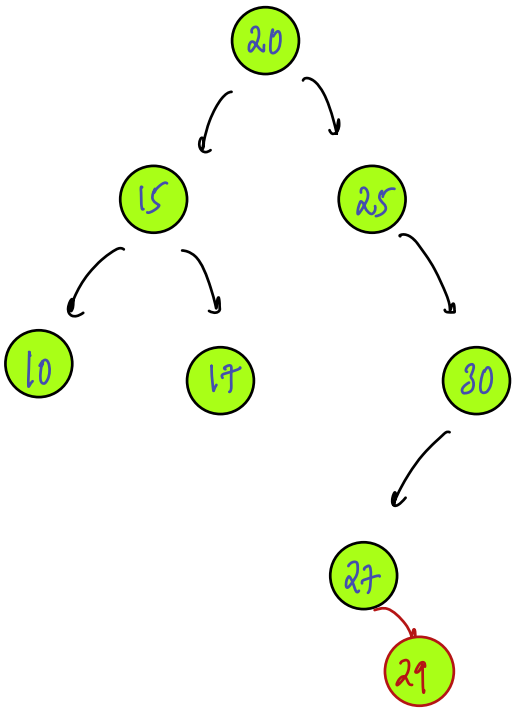
Best: H = $\log_2^N$? (At last)

SC: O(1)

// Insert in BST: → 29 : {TODO}

Iterative:

Recursive:



// Tuesday: → Construction: All Construct Problem

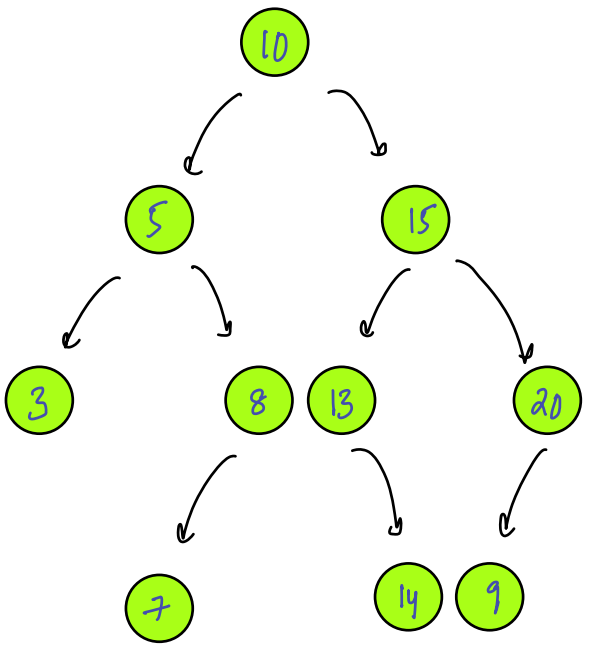
1) Inord / pr →
2) BT → CDLL
3) BST → SCDLL
4) array → BBST

//

Q8) Given BT check BST or not?

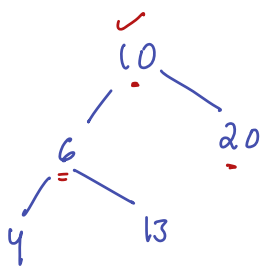
Idea1: Store inorder & check sorted or not

TC: O(N) SC: O(N)



Idea2: For every node check with

root.left.data < root.data < root.right.data



Idea3: For all Nodes [All elements in LST < node < All elements in RST]

LST → max

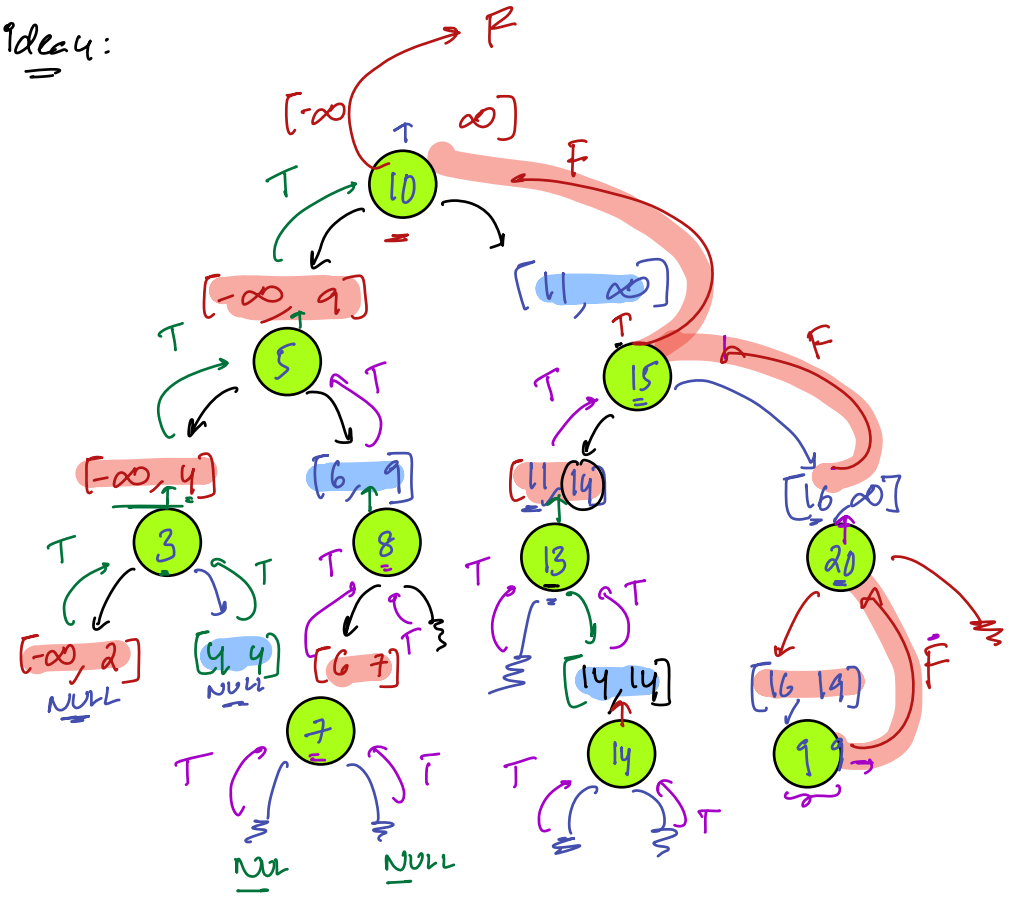
RST → min

TC: N * { N + N } → O(N²)

get max in LST

get min in RST

Idea:



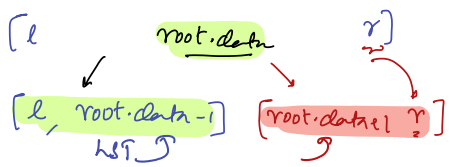
// Ass: → given root node, check if all node are in [l r]

bool isBST(Node root, int l, int r) { → TC: O(N)

if(root == NULL) {return True}

if(root.data >= l && root.data <= r){

return



(isBST(root.left, l, root.data-1) &&

isBST(root.right, root.data+1, r))

}

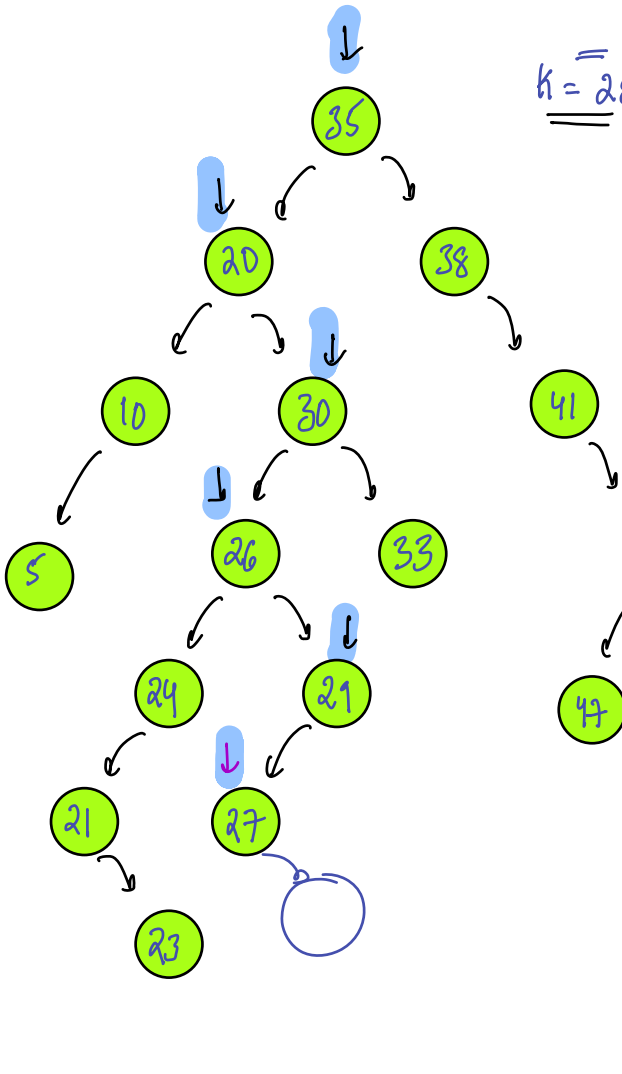
else : return False

}

→ Given k, find floor of k in BST

→ greatest element ≤ k in BST.

Eg:



k = 28

35 > 28 : cannot, goto left

20 < 28 : possible : ans = 20
goto right

30 > 28 : cannot, goto left

26 < 28 : possible, ans = 26
goto right

29 > 28 : cannot, goto left

27 < 28 : possible, ans = 27
goto right

NULL :

ans = 27

```
int flor(Node root, int k){
    ans = -∞ / INT_MIN
    while(root != NULL){
        if(root.data == k)
        If(root.data > k){
            //cannot go left
            root = root.left
        }
        else if(root.data < k)
            // possible; ans = root.data
            // Better; root = root.right
    }
    return ans;
}
```

k-1

// flor of k = greatest <= k

{ return k; : flor <= k
root = root.left : flor < k }

TC: O(H)

SC: O(1)

TODO: — ceil(Node root, int k) {

1) smallest element > k

2) smallest element >= k

}

TC: O(H) SC: O(1) → { 10:52 pm }

Story: Barley ki Barfi

→ (She is Barley)

IIT/NIT not well

(gone)

Bad Rank

Seat in Govt College → {2/3 per year}

→ Engineering: ECE

→ (2 years): Wasted

→ (3rd: gate - 4th) gate:

→ not well

→ not good rank

IIT hyderabad ⟶ IIT Delhi

: She got it

: No you cannot go

: yy

: IIT → CSE

: no coding background

: 2 years → 1 year: had work & gpg

2nd year: Placements

Placements

→ Microsoft

→ Snapdeal / Amazon

→ Google: (last round)

→ Sholom: (Job)

1 years →

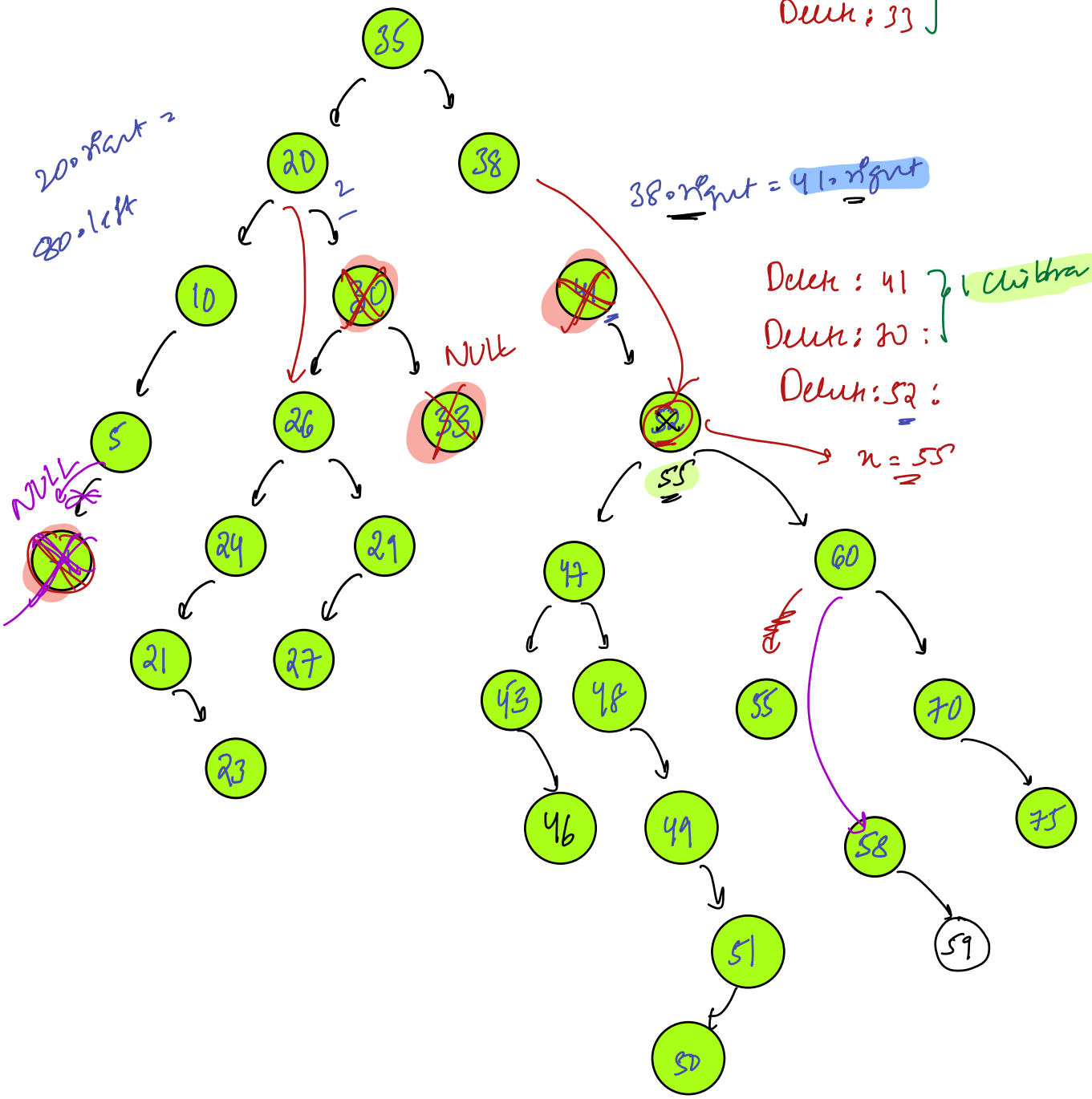
Amazon → Higher Stab in Sde1

Goldman → SDt2 ✓

Scalar Instructor: Shubrattu

BST: → Delete k in BST → k is present

Delete: -1
Delete: 45
Delete: 33
} no children



Delete: 41
Delete: 30:
Delete: 52:
} 1 children

Ass: given root node, delete k in BST, & return new Root

```
Node delete(Node root, int k){
    if(root == NULL){ // Element not there
        return null
    }

    if(root.data == k){
        // Case: I →
        if(root.left == NULL && root.right == NULL){
            return NULL
        }
        // Case: II →
        if(root.left == NULL || root.right == NULL){
            if(root.left == NULL) { return root.right }
            else { return root.left }
        }
        // Case: III → use: max in LST or min in RST
        {
            int x = min(root.right)
            root.data = x
            root.right = delete(root.right, x)
            return root;
        }
    }
```

```
if(root.data > k) { // goto left & delete
    root.left = delete(root.left, k)
}
else if (root.data < k) { // goto right & delete
    root.right = delete(root.right, k)
}
return root;
```

root of LST

root of RST

TC: O(H)

BT / BST

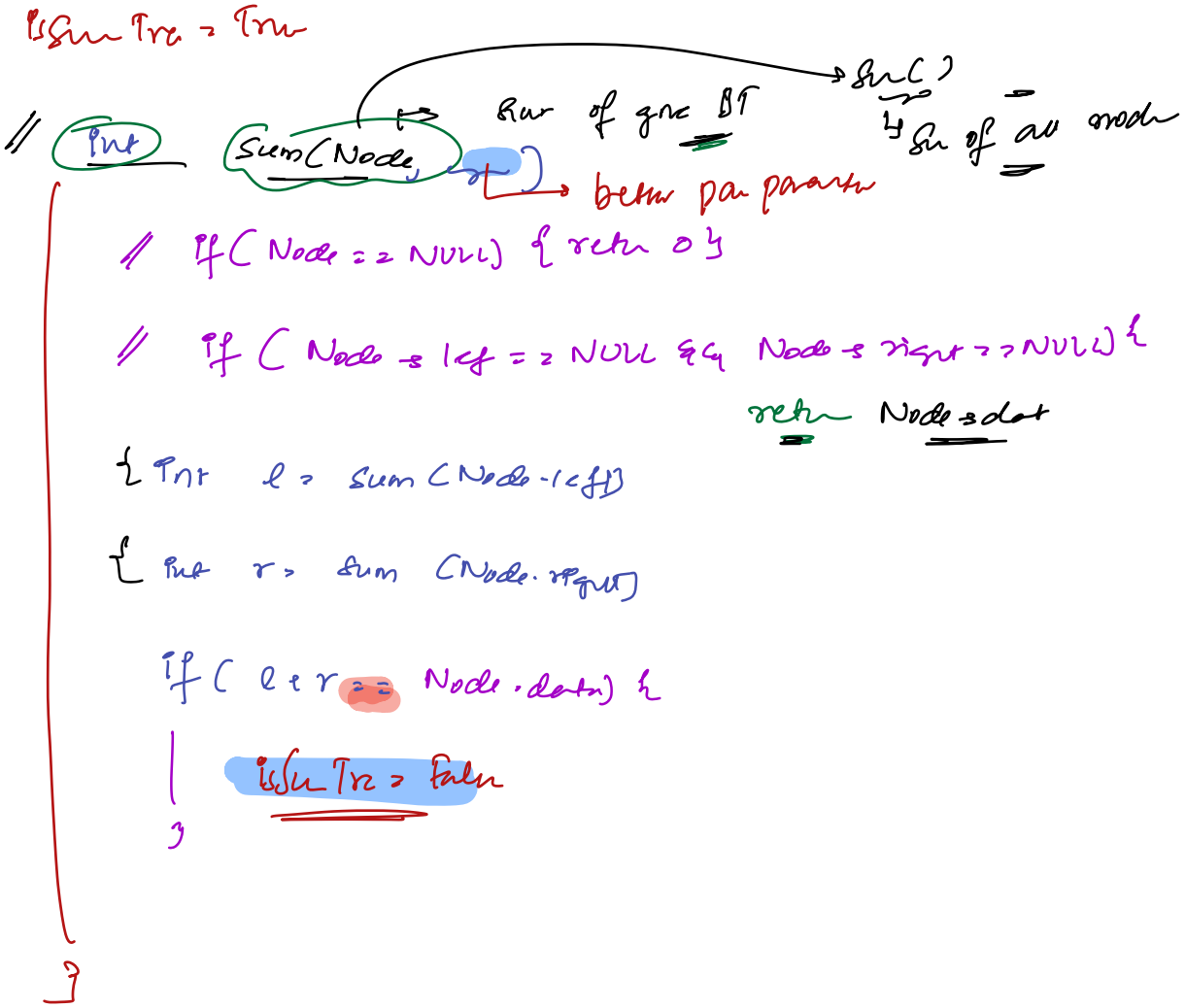
↳ 3 more sessions:

Some many problems

TreeSet / TreeMap

Sort → Sum → Min → Max

Doubt:


isSumTree = True
int
sum(Node)
Sum of given BT
sum()
Sum of all nodes
better parameter
// if(Node == NULL) { return 0 }
// if (Node->left == NULL && Node->right == NULL) {
return Node->data
{ int l = sum(Node->left)
{ int r = sum(Node->right)
if (l + r == Node->data) {
isSumTree = false
}
}