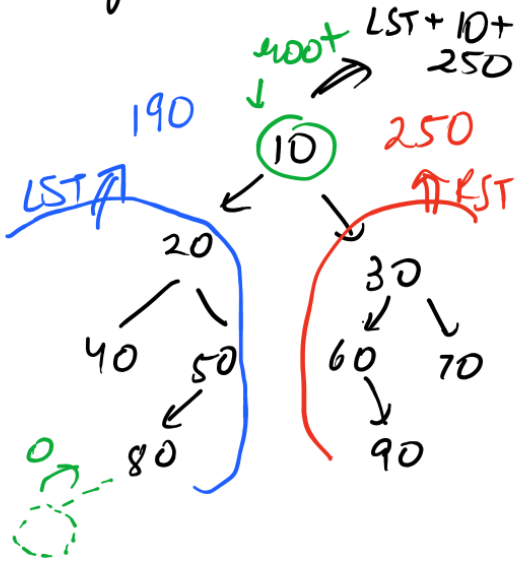
From: **Mihir Sood** mihirsood.mait@gmail.com
Subject:
Date: 2 August 2024 at 12:45 AM
To:

MS

Binary Trees-2

Thursday, 1 August 2024 8:50 PM

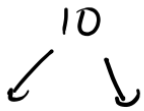
3) Sum of a tree → {Return the sum of data of all the nodes present in the tree}



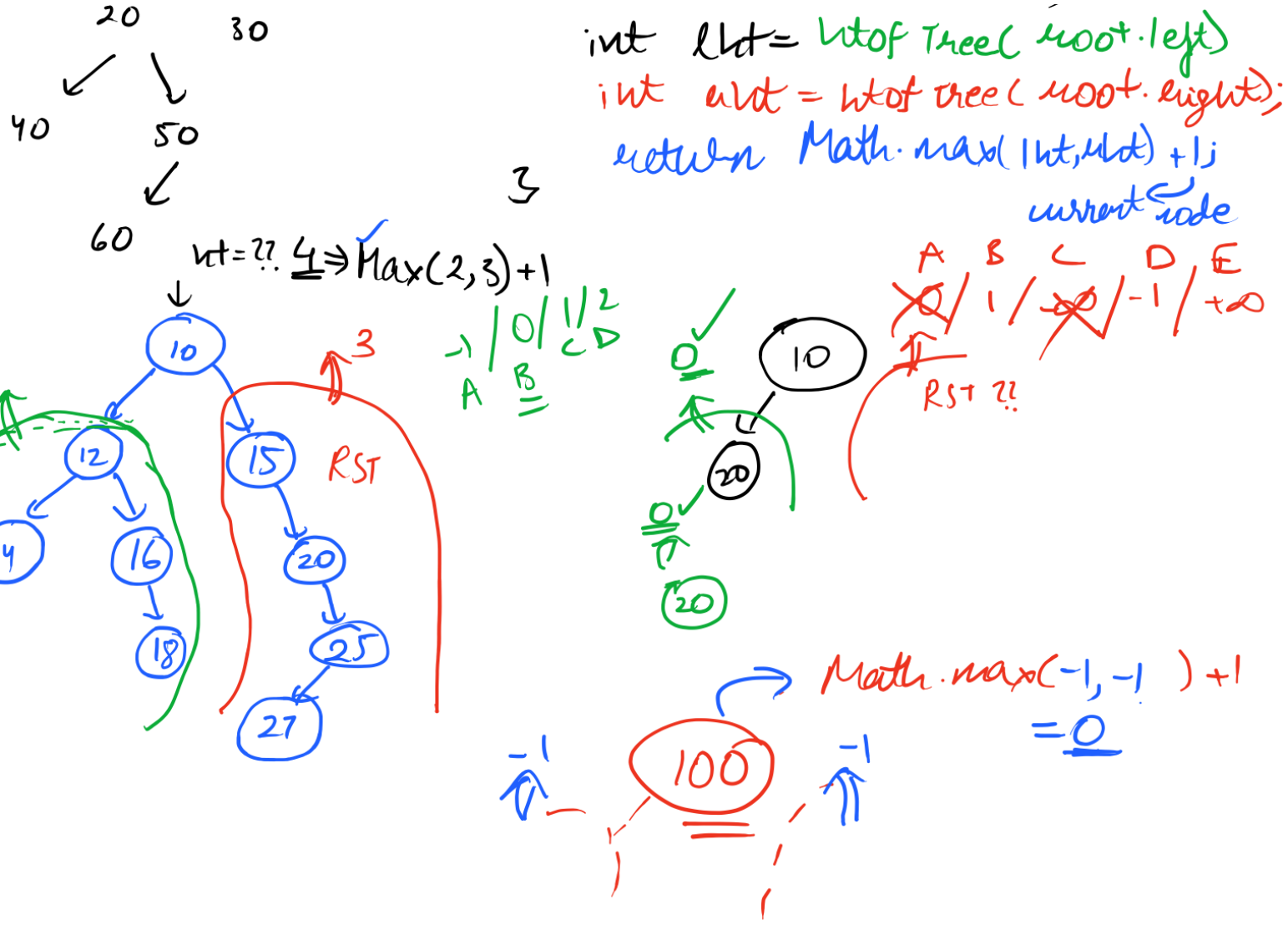

```
int sumOfTree(Node root){
//B.C -> root==null return 0;
  int lans = sumOfTree(root.left);

  int rans = sumOfTree(root.right);
  return root.data + lans + rans;
}
```

4) Height of the tree ⇒ The distance b/w root and the deepest node. ⇒ On the basis of edges.



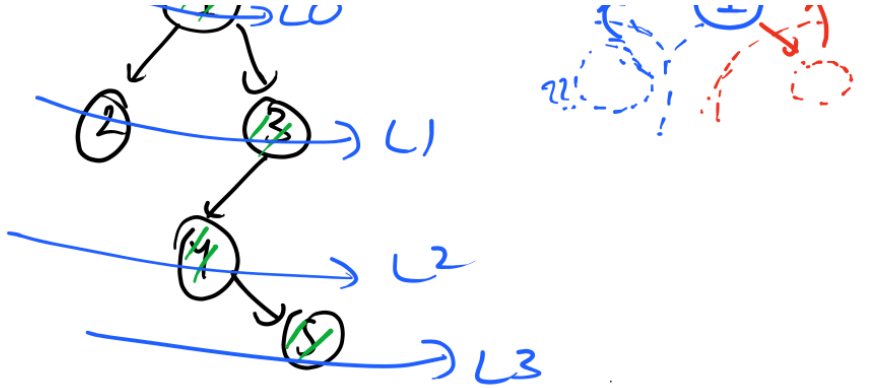

```
int ht of Tree(Node root){
  if(root == null) return -1;
```



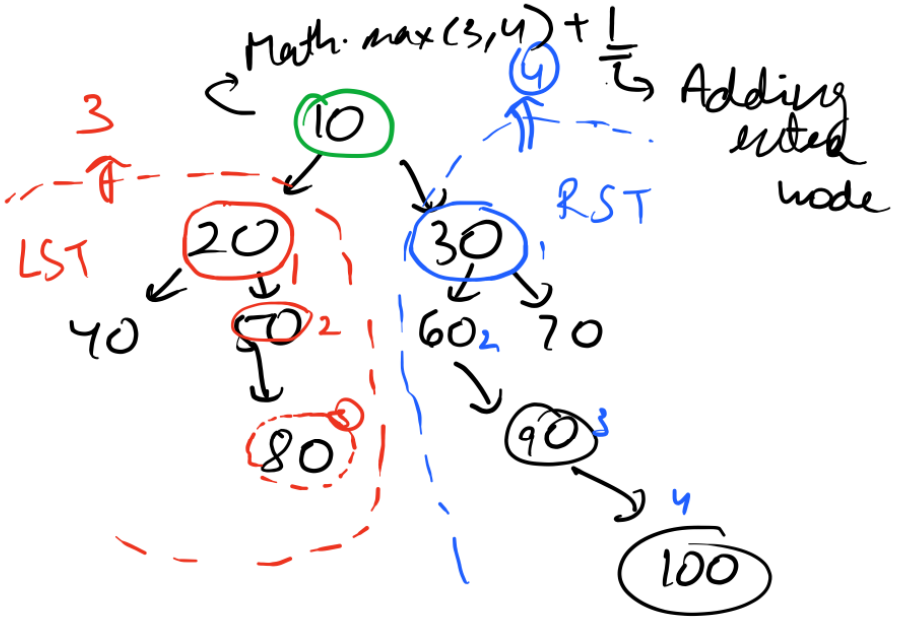
Ht of the tree on the basis of nodes?

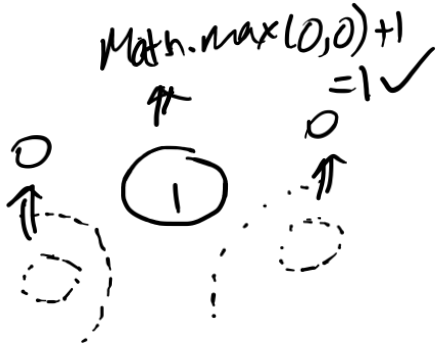
Math.Max (ans, rans)+1 → No. of nodes

0+1



present b/w [root node and deepest node]



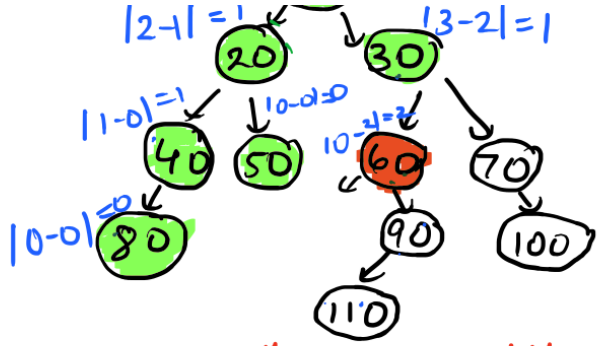


Balanced Binary Tree

|3-4|=1

Balanced Node →

|ht LST - ht RST| ≤ 1



NOT BALANCED
return false.

↪ the tree with root is Balanced or not

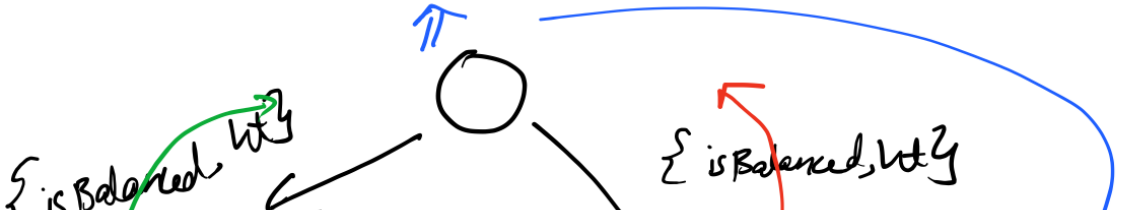
```
boolean isBalanced (Node root){
  if (root == null) return true;
  int lht = ht of tree (root.left)
  int rht = ht of Tree (root.right)
  int abs = Math.abs(lht - rht);
```

O(N²) → O(N) for N nodes

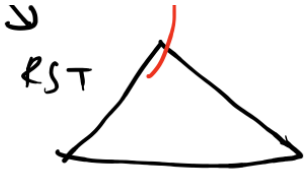
↩ O(N) Checking if curr node is Balanc

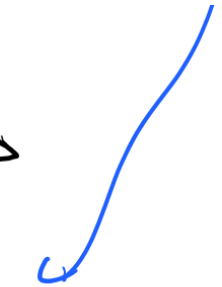
```
  boolean isLeftBalanced = isBalanced (root.left); //Checking L
  boolean isRightBalanced = isBalanced (root.right); //Checking
  return (abs ≤ 1) && isLeftBalanced && isRightBalanced
}
```

Better ??



⇒ O(N)





{ isRootBalanced, ht Root }
↳ Math.max(lht,

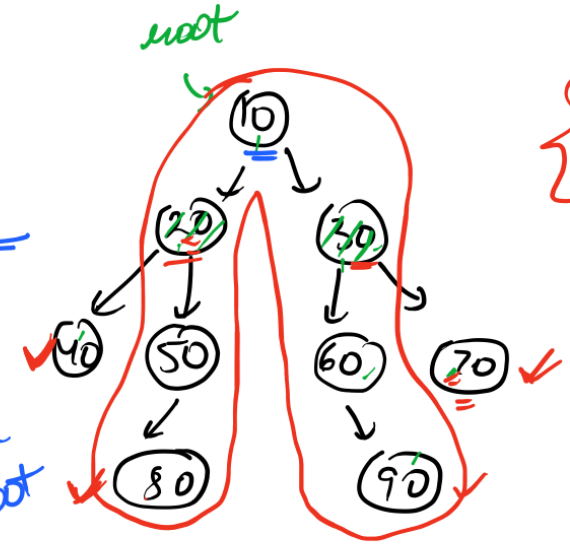
lht, rht
isLeft, isRight

Math.abs(lht - rht) ≤ 1
&& is LeftBalanced
&& is RightBalanced

isBal → |0-0| = 0 ≤ 1 ✓ ⇒ true
ht → (0,0) + 1 = 1 ✓ ⇒ 1

isBal → |0 - 0| ⇒ true
ht ⇒ 0

Diameter of a Binary Tree = { Maximum Distance b/w any 2 nodes of a tree } ⇒ in terms of nodes



Diameter is passing through the root

⟹ will be leaf nodes.

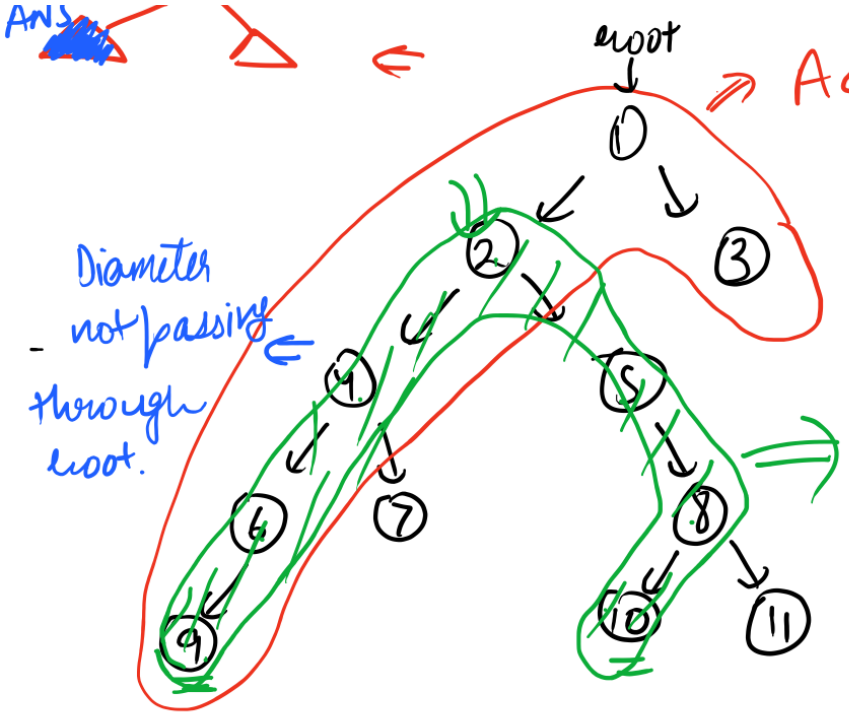
dist(20,30) = 3
dist(30,40) = 4
dist(40,70) = 5
dist(90,70) = 4

Claim: Distan... will be max... when consi... leaf nodes

dist(40,80) = 4
dist(40,70) = 5
dist(40,90) = 6
dist(80,70) = 6
dist(80,90) = 7 ⇒ Ans

diameter = { ht LST + ht RST + 1 } ✗ ✓ Wrong

Note: Diameter might not always pass through root



Diameter not passing through root.

According to above formula
$= 4 + 1 + 1 = 6$
↳ Wrong Ans

dist (9, 10) = 7 ⇒ Corr Ans

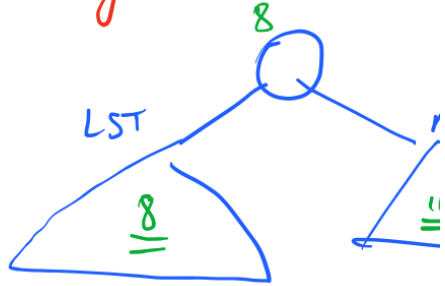
Note: diameter may not always pass through the

{ ht LST(2) + ht RST(2) + 1 }

↳ faith: returns the diameter of tree starting with root

```
int diameter (Node root) {
if(root==null){return 0};

int LSTht= ht (root.left);
int RSTht = ht (root.right);
```



```
int diameterWithRoot = LSTht + RSTht + 1;
```

- When diameter is not passing through root

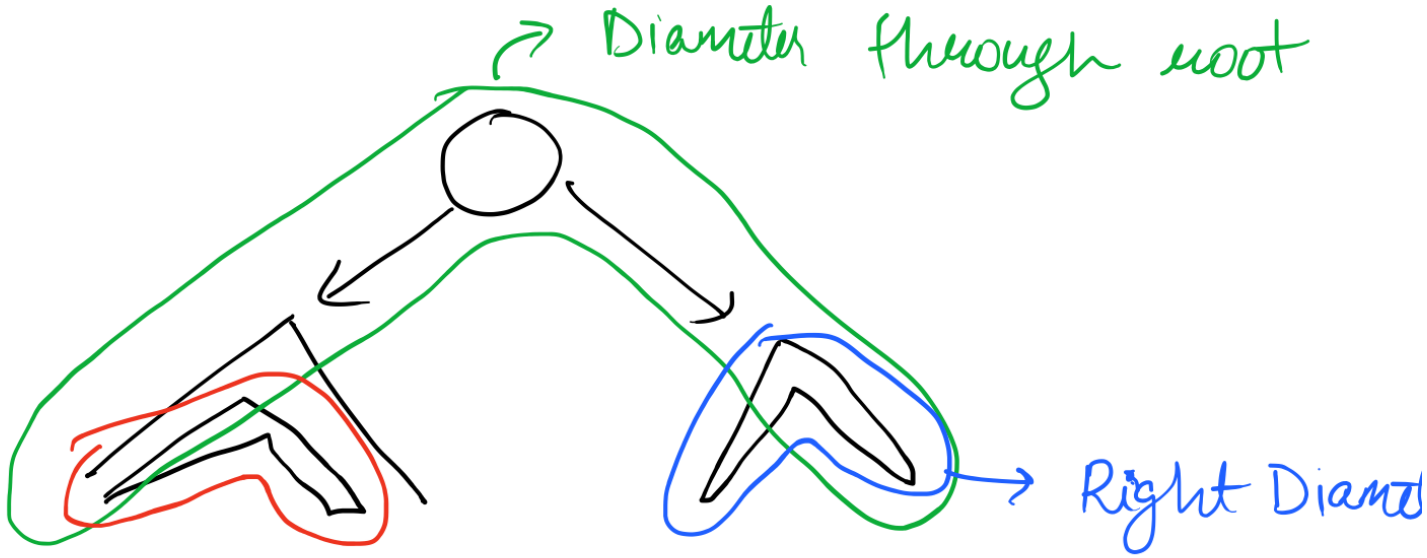
```
{ int diameterFromLST = diameter(root.left);
  int diameterFromRST = diameter(root.right);
```

```
return Max{ diameterWithroot, diameterRST, diameter
```

-

```
} ⇒ TC: { N × O(N) = O(N²)
```

Optimise?!



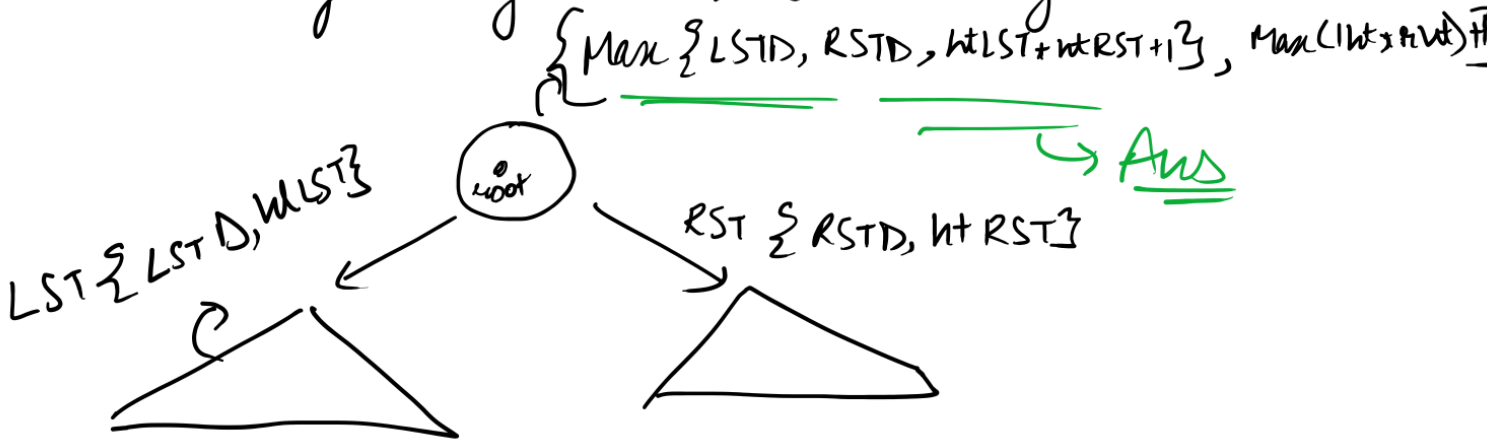
Left Diameter

Ans = Max{ Diameter through root, RST Diameter, LSTD

Optimisation

↳ We require both the ht as well as the best possible diameter from left and right.

⇒ Instead of making 2 calls, make a single call.



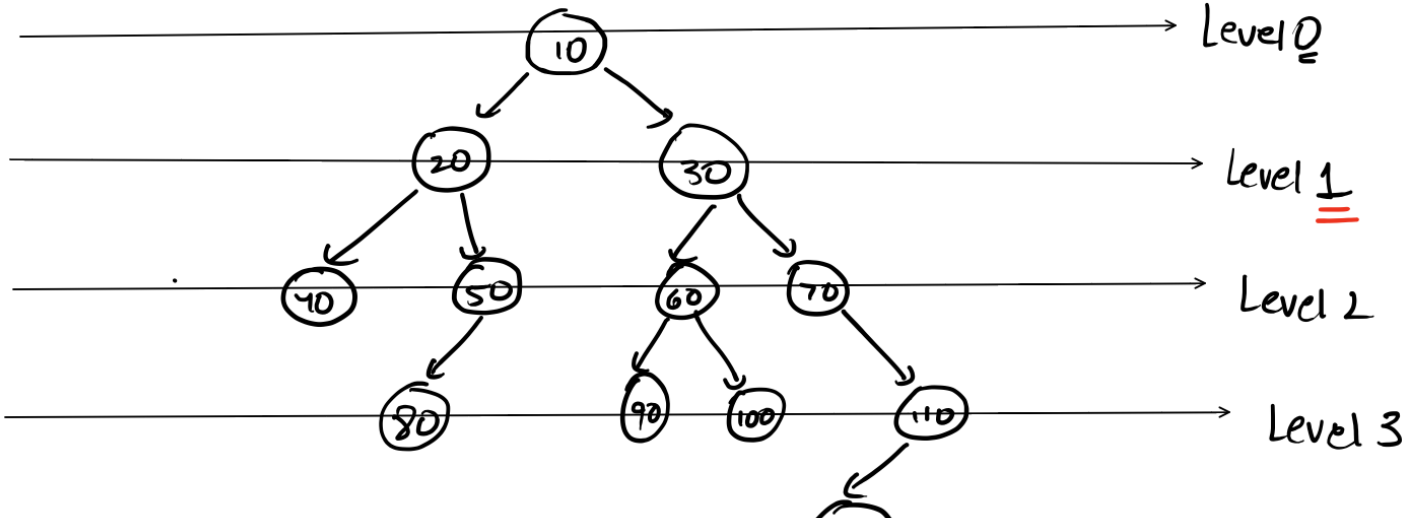
Base case

TC: Σ O(N) → Every Node will be called only once

SC: O(H)
↳ Ht of the tree.

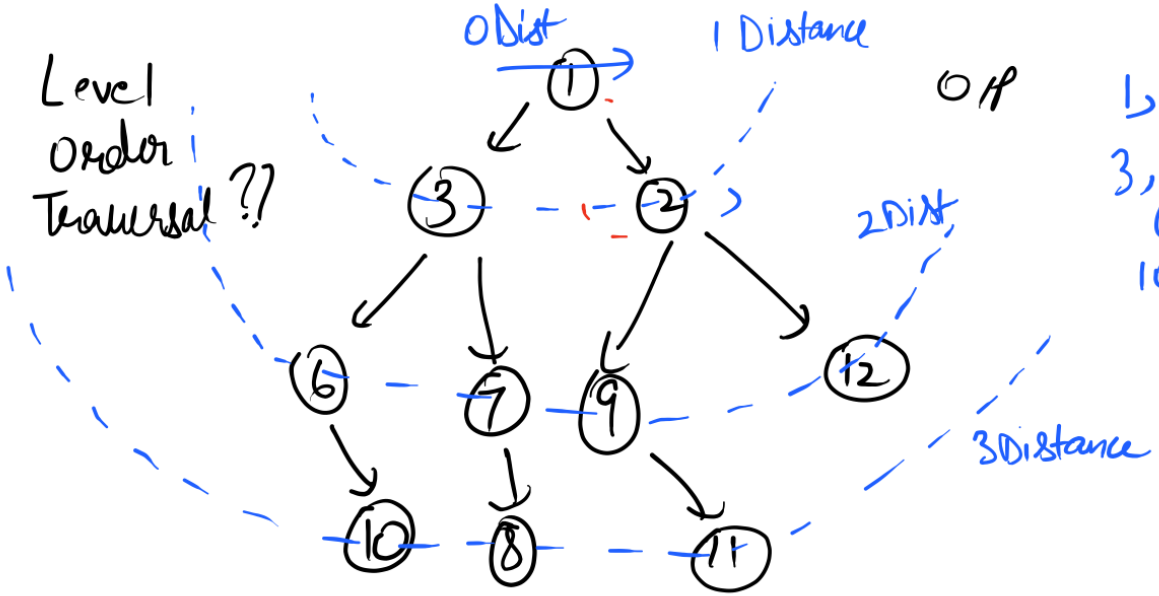
—

Level Order Traversal



(120) → Level 4

O/P: 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120

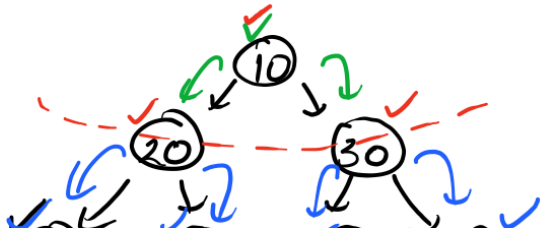


Level Order Traversal ??

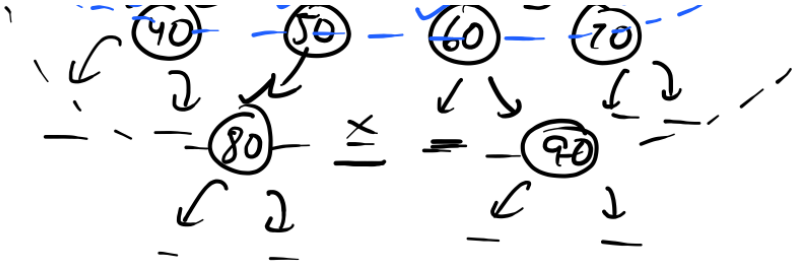
O/P 1, 3, 2, 6, 7, 9, 12, 10, 8, 11

traversing radially ⇒ BFS

↳ We require Queue ✓



10 20 30 40 50 60 70 80 90



size = 10 20 40 80 160

10, 20, 30, 40, 50, 60, 70, 80, 90 → level

TC: O(N) → No. of Nodes

SC: → O(2^k)

↳ At a given time all the elements from a particular level will be present in the queue.