Trees 5

Rajat Sharma
sharath r
Saurabh Ruikar
Khushi Raj
Ishan
Rajendra
Subhranil Kundu
Vasanth
amit khandelwal
Madhan Kumar M S
Subhashini
Sneha L
Pankaj Bhanu
Sumit Adwani
PREETHAM
Vimal Kumar
Burhan
Gowtham
Suyash Gupta
Sanket Giri
Bhaveshkumar
Purusharth A
Prajwal Khobragade
Nikhil Pandey
Gagan Kumar S
Hemant Kumar
Shradha Srivastava
Abhishek Sharma

GREAT JOB!

AGENDA:

Invert Binary Tree

Equal tree Partition

Next Pointer in Binary Tree

Path Sum equals K

Diameter of Binary Tree

$$\begin{tikzcd}\text{Current PSP} \arrow[d] & \\ 62 \arrow[r] & 70\% \end{tikzcd}$$

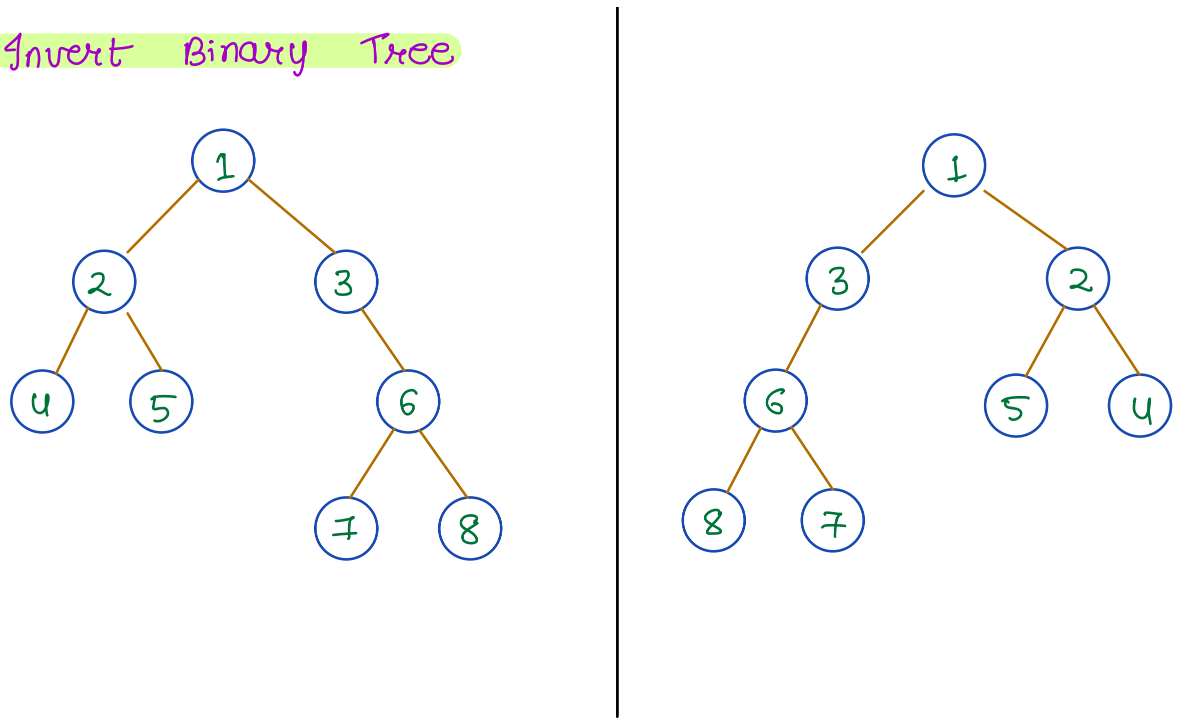
Important Announcement

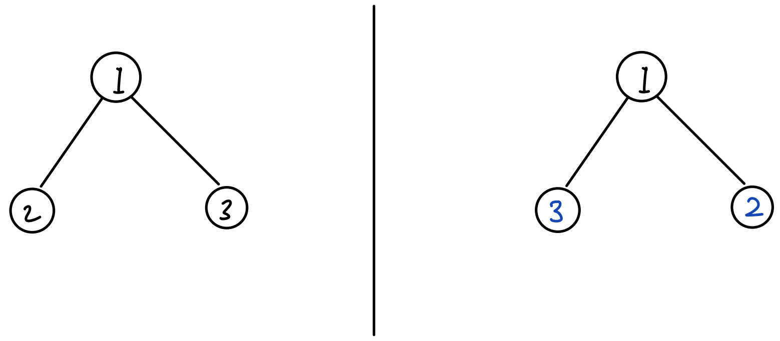
Contest 4 $\longrightarrow$ 5th April

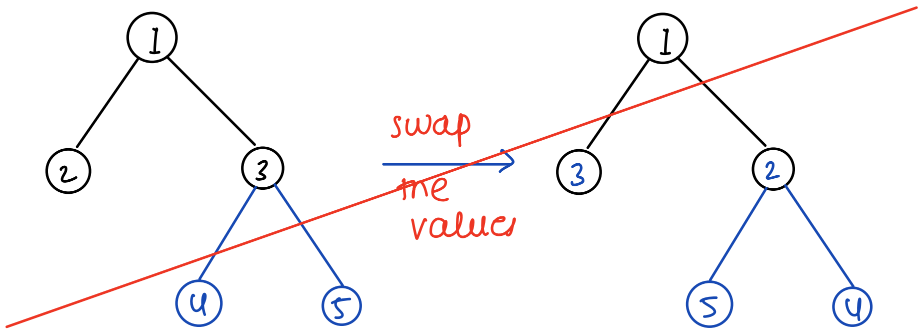
Two pointers, LL, Stacks & Queues

Google

Invert Binary Tree







⇒ Swap the left and right at each node

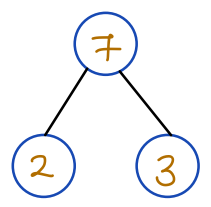
Pseudocode

```
                                        // Preorder
void   invert ( TreeNode root) {
    if (root == null)   return

    // swap left and right
    TreeNode temp = root.left
    root.left = root.right
    root.right = temp.

    invert( root.left )
    invert ( root.right )
}

main {
    invert(root)                        TC: O(N)
    return root                         SC: O(H)
}
```

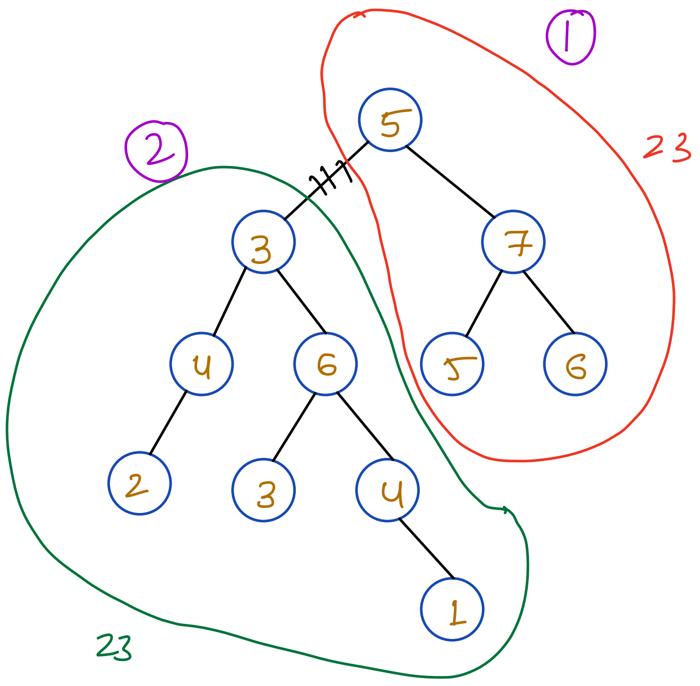
Equal tree Partition

Check if it is possible to remove an edge of binary tree such that

sum of the resultant two trees is equal



ans = false



ans = true

Hint 1 $\longrightarrow$ A subtree is formed by breaking an edge.

Hint 2 $\longrightarrow$ sum { one tree } == sum { two tree }

$\Rightarrow$ sum { subtree } == half of overall tree

Pseudocode

```
int sum ( root ) {
    if ( root == null )  return 0

    return  root.val  +  sum(root.left) +
                         sum (root.right)
}

total  =  sum(root)
if ( total % 2 == 1 )  return  false
```

```
// Bruteforce idea  ∀ nodes  we  above  sum  ==  total/2
partition  =  false

void  inorder ( root ) {
        if ( root == null )  return

        inorder (root.left)                    TC:  O(N^2)

        subTotal  =  sum (root)
        if (subtotal  ==  total/2 )
                partition  =  true

        inorder (root.right)
}
```

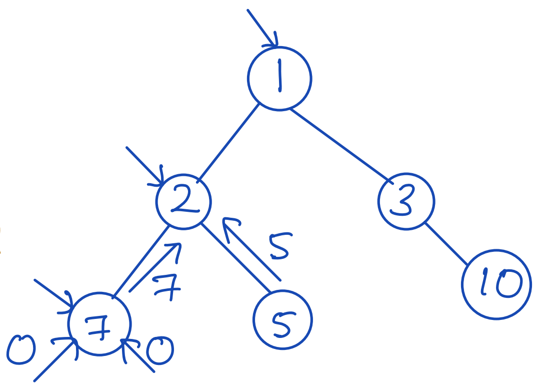
Optimal

```
int sum ( root ) {
    if ( root == null )  return 0

    return  root.val + sum(root.left) +
                       sum (root.right)
}
```

```
total = sum(root)                          TC : O(N)
if ( total % 2 == 1 )  return  false       SC : O(H)
partition = false
int   postOrder ( root ) {
      if ( root == null )  return 0

      ltotal = postorder ( root.left )
      rtotal = postorder ( root.right )
      ctotal = root.val + ltotal + rtotal

      if ( ctotal == total/2 )  partition = true

      return ctotal
}
```

total = 28

$total/_2$ = 14

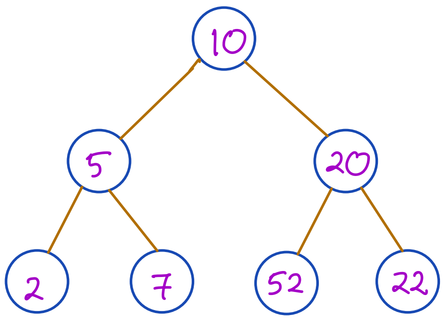


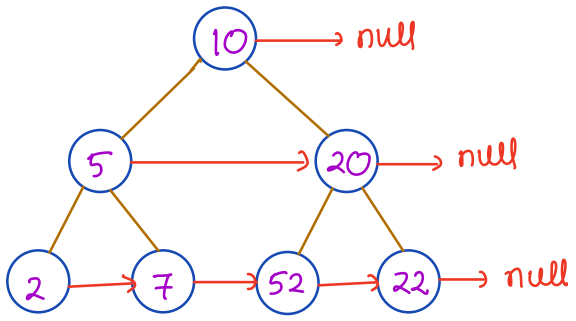
Next Pointer in Binary Tree

Given a perfect binary tree.
Initially ∀ nodes, next pointer is NULL
update next pointer to point next node in same level {left to right}

```
class Node {
        left
        right
        next
        val
}
```

Input



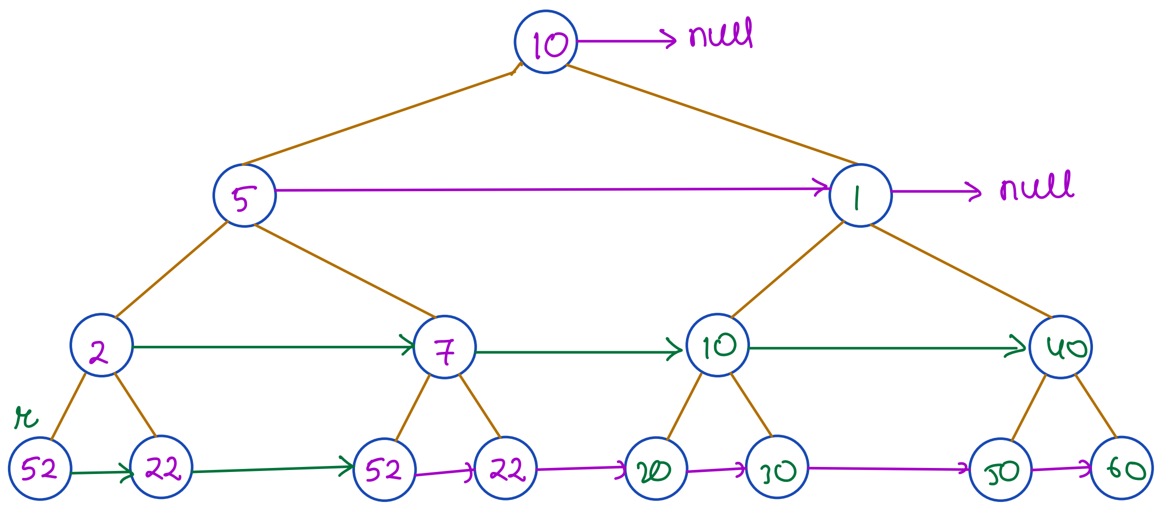


Idea 1

use level order traversal and join all the nodes as a linked list using next pointer in the same level

TC : O(N)

SC : O(N) $\rightarrow$ O(1) optimise



Pseudocode

TC : O(N)
SC : O(1)

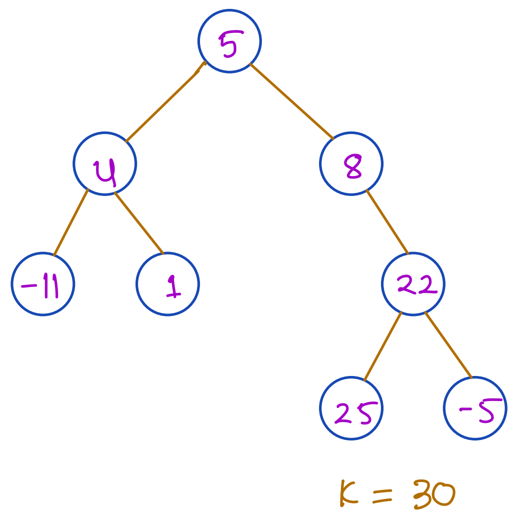
```
r = root
while ( r != null  &&  r.left != null ) {
        temp = r
        // Connect next of entire level
        while( r != null) {
            r.left.next = r.right
            if( r.next != null)
                    r.right.next = r.next.left
            r = r.next
        }
        r = temp.left
}
```

Extend the above approach for normal BT.

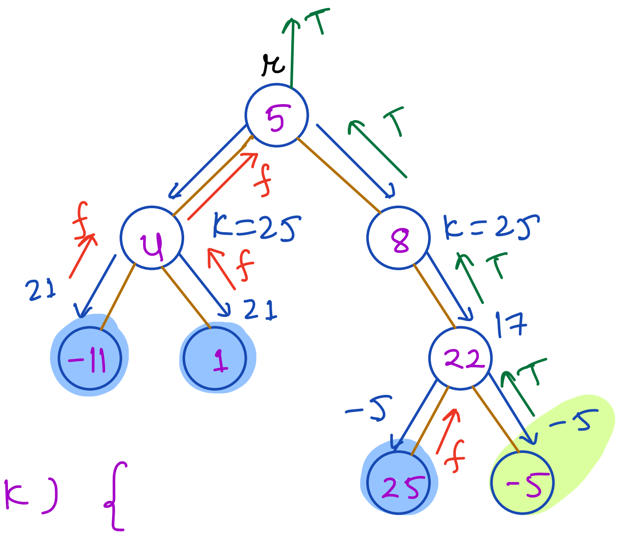
22:42

Path sum equals K

Check if any root to leaf path sum equals K



K= 13	ans = false
K= 60	ans = true
K= 50	ans = false
K= 30	ans = true



TC: O(N)

SC: O(H)

Pseudocode

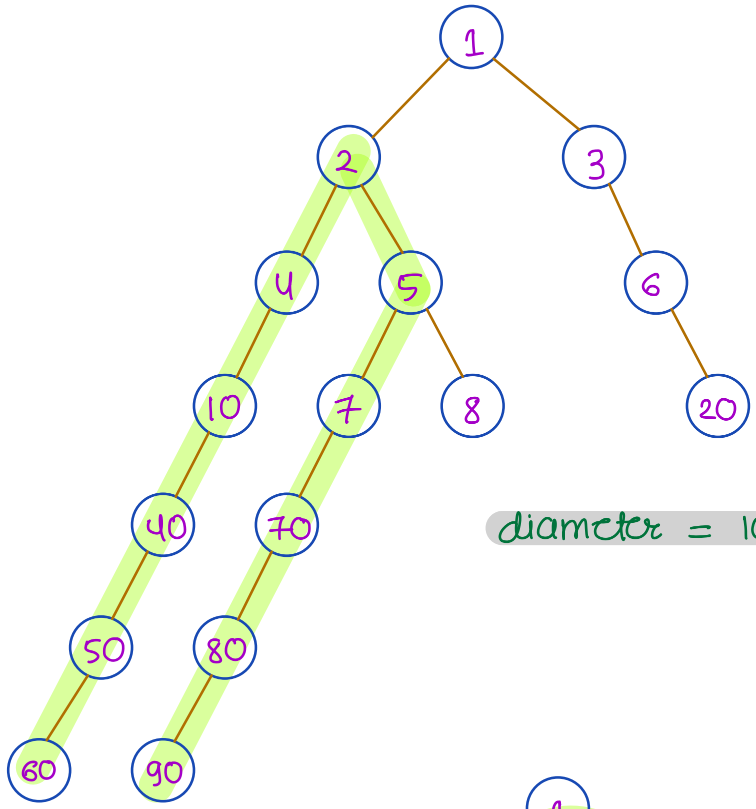
```
boolean hasPath ( root , k ) {
        if( root == null) return false
        // leaf condition
        if (root.left == null && root.right ==null) {
            return root.val == k
        }
        return hasPath(root.left, k - root.val) ||
                   hasPath(root.right, k - root.val)
}
```

Diameter of Binary Tree

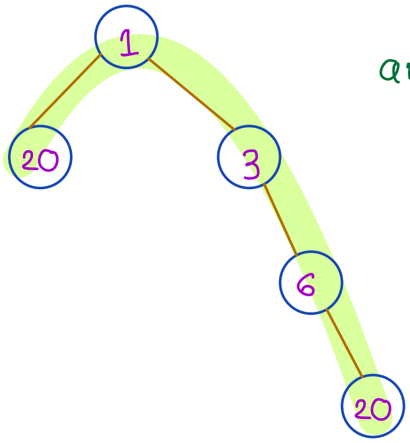
{ Maximum distance b/w any two leaf nodes in a binary tree }

in terms of no. of edges

Longest path through any node in the tree



diameter = 10



ans = 4

How to calculate height of a tree ?

Bruteforce

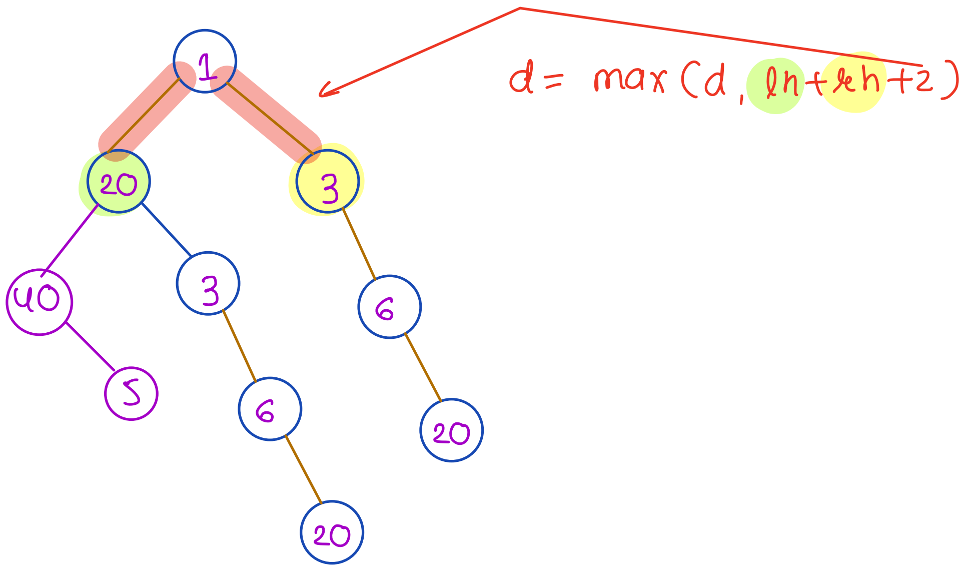
d=0

```
int height ( root) {
    if ( root == null)  return -1

    lh = height (root.left)
    rh = height (root.right)

    return max(lh, rh) + 1
}
```

∀ nodes calculate lh & rh using above height fn"

TC: $O(N^2)$
SC: $O(H)$

d= max(d, lh+rh+2)



Pseudocode

```
                    d=0
int height ( root) {
    if ( root == null)  return  -1

    lh = height (root.left)
    rh = height (root.right)
    d = max (d, lh + rh + 2)
    return max (lh, rh) + 1
}
```

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

How to do without global variable ?

```
int height ( root , d[] ) {
    if ( root == null)  return  -1

    lh = height (root.left , d[] )
    rh = height (root.right , d[] )
    d[0] = max (d[0], lh + rh + 2)
    return max (lh, rh) + 1
}
```

```
main {
        d = new int [1]
        d[0] = 0
        height ( root , d[]  )
        print(d[0])
}
```