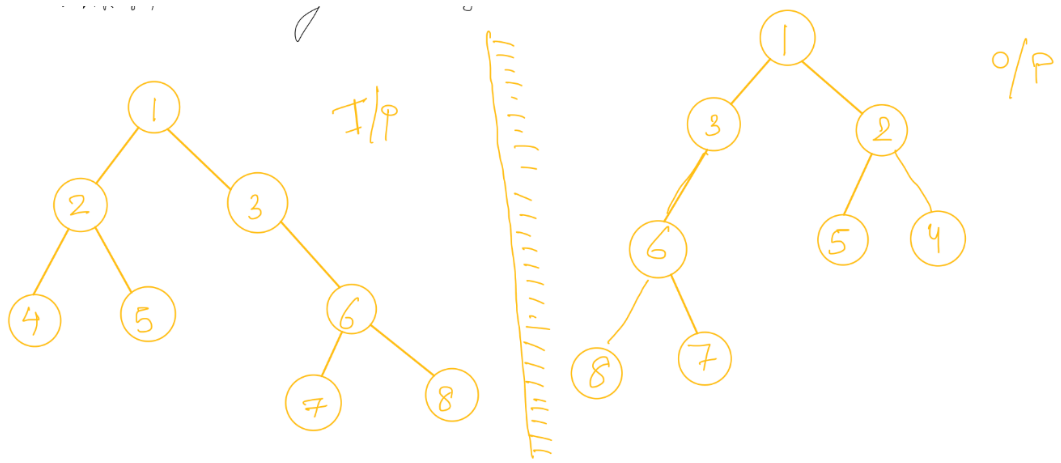
Trees 5

Invert Binary tree



Pseudo code

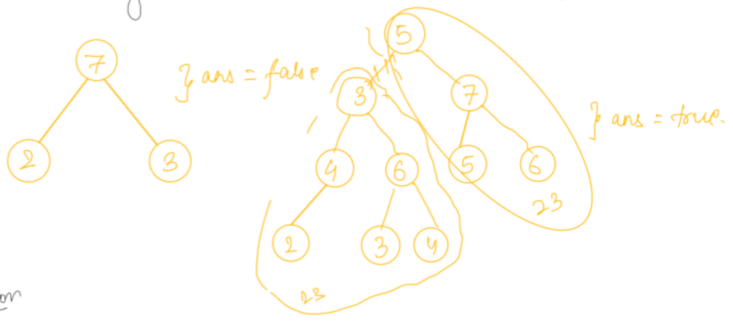
```
void invert (Node root)
{
    if (root == nullptr)
        return;

    //swap the left and right child.
    Node temp = root.left;          }
    root.left = root.right;         } swap
    root.right = temp;              }
    invert (root.left);
    invert (root.right);
}
                  //preorder
```

$\boxed{\begin{array}{l} T.C : O(N) \\ S.C : O(H) \end{array}}$

Equal Tree Partition

Q. Check if it is possible to remove an edge from binary tree. Such that sum of resultant two trees is equal.



Observation

$$\begin{tikzcd} 20 \arrow[r] & \text{Item1 (10)} \\ & \text{Item2 (10)} \end{tikzcd}$$

23 → Not possible to distribut

So, if total sum of all nodes is odd ⇒ ans = false
if total sum of all nodes is even ⇒ ans = check?
↳ check if there is a subtree with sum = Total Sum/2.

Pseudo code:-

```
int sum (Node root)
{
    int l = sum (root.left)
    int r = sum (root.right)
    Return root.data + l + r;
}
int totalSum = sum(root);
if (totalSum % 2 == 1)
        return false;
```

```
//check if there is a subtree with sum = total sum/2;
int sum (Node root)
{
    if (root == NULL)
        return 0;
    int l = sum (root.left);
    int r = sum (root.right);
    if (l == total sum/2 || r == total sum/2)
        return 1; "true"
    return root.data + l + r;
}
```

check what is returned by sum function…..

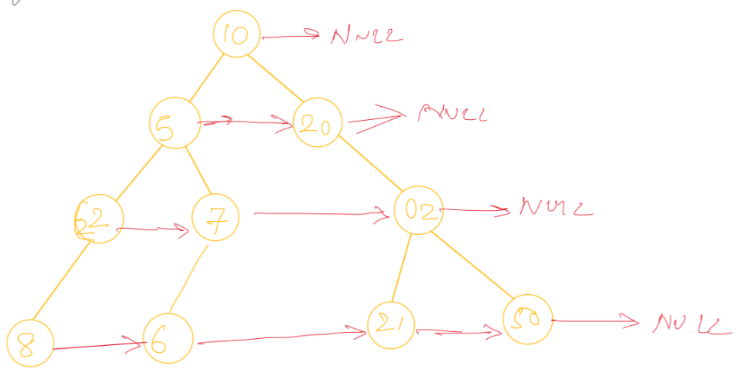
Next pointer in binary tree

Q Next pointer in binary

Initially ∀ nodes, next pointer points to NULL. Update next pointer to point to next node in same level. (left to right)

eg:

Observation? level order



Idea:
Simple level order traversal.

~~10~~ (N) ~~5~~ 20 ~~N~~

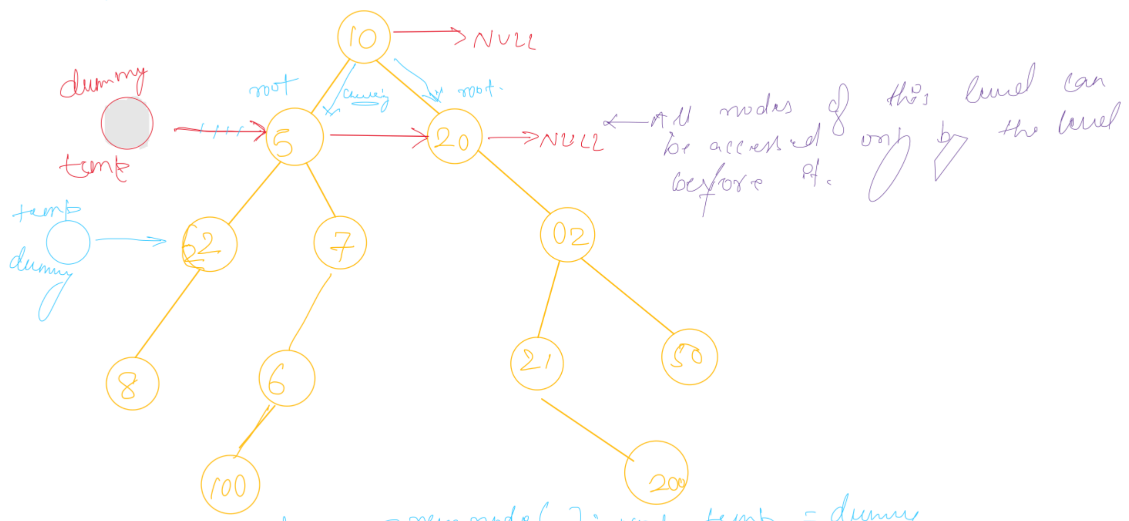
f

```
if(f==NULL)
{
    q.pop()
    q.push()
}
else{
    int f = q.front()
    q.pop()
    f.next = q.front()
    q.push(f.left)
    q.push(f.right)
}
```

T.C: O(N)
S.C: O(Max nodes in a level)

Idea 2 :-

To check again


10
NULL
dummy
temp
root
curr
root
5
20
NULL
All nodes of this level can be accessed only by the level before it.
temp
dummy
2
7
02
8
6
21
50
100
200

Node dummy = new node( ); Node temp = dummy

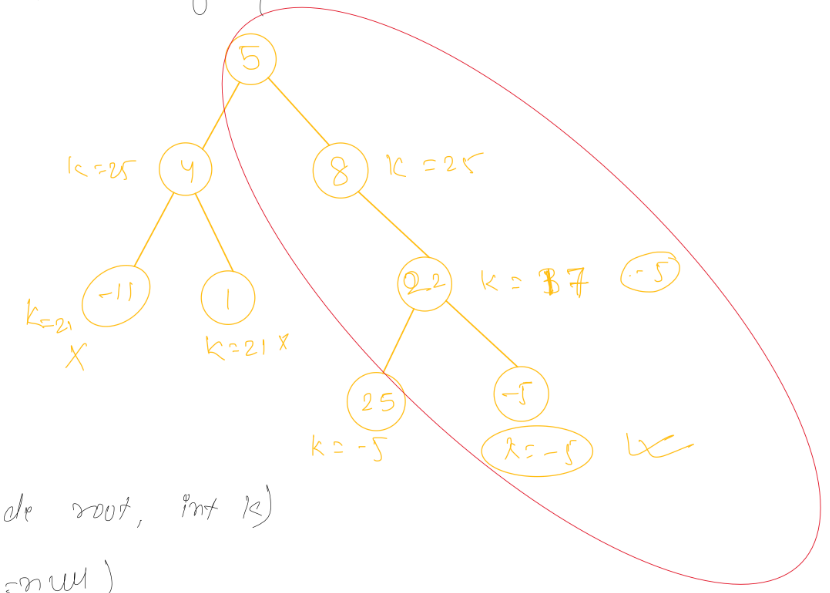
Pseudo code :-

```
while (root != NULL) {
    if ( root.left != nullptr) {
        temp.next = root.left;
        temp = temp.next;
    }
    if (root.right != nullptr) {
        temp.next = root.right;
        temp = temp.next;
    }
    root = root . next;
    if ( root == null ptr)
        root = dummy. next;
        dummy. next = nullptr;
        temp = dummy;
    }
}
```

Path sum equal k

Q- Check if any root to leaf path sum equals k.

k = 60 : true
k = 50 : false
k = 30 true



```
bool Check (Node root, int k)
{
    if (root == null)
        return (k == 0); // for empty tree & k==0
    if (root -> left == NULL && root -> right == NULL)
    {
        return root.data == k;
    }
    return check (root.left, k - root.data) ||
           check (root.right, k - root.data)
}
```

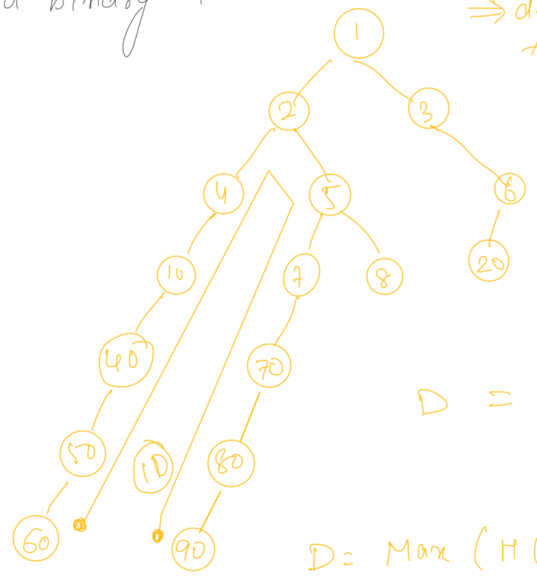
T.C : O(N)
S.C : O(H)

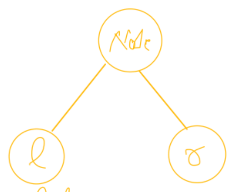
Diameter of a tree

Q Diameter of a Binary Tree

⇒ Maximum distance between any two leaf nodes in a binary tree

⇒ diameter need not to pass through root always.





D = Height of l + height of r

D = Max (H(l) + H(r) + 2) ∀ nodes

```
int Height(Node node)
{
    if (root == null)
        return -1;
    int l = height (root.left);
    int r = height (root.right)

    ans = max(ans, l+r+2); //diameter.
    return max(l, r) +1
}
```