TREES 5

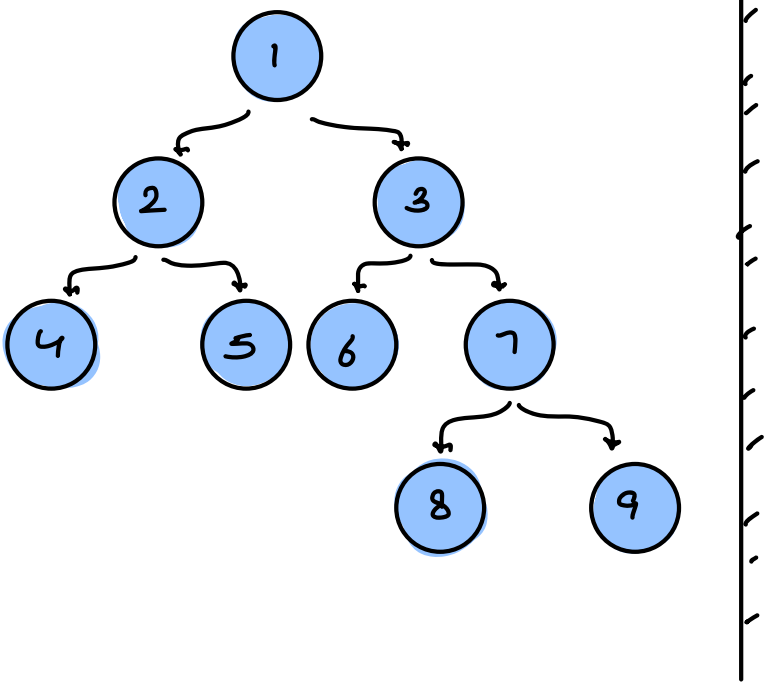
Good Evening ☺

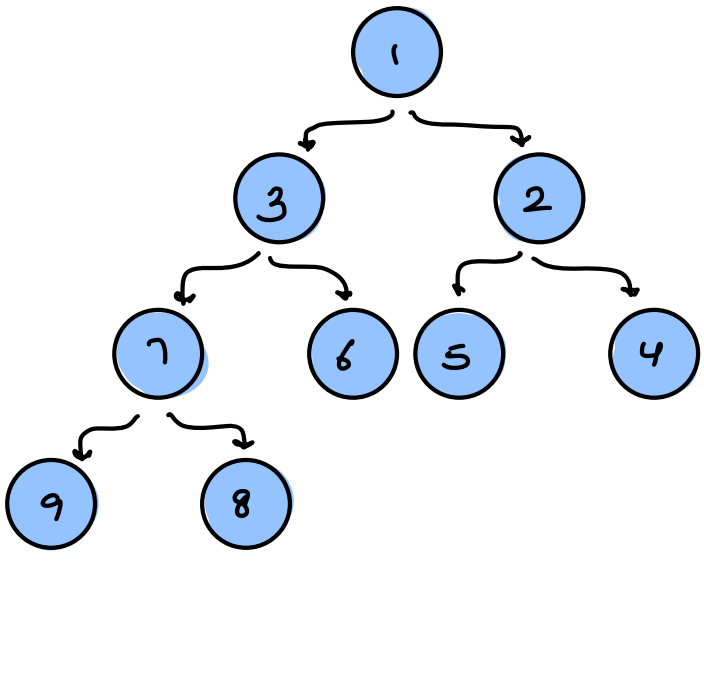
* Today's content

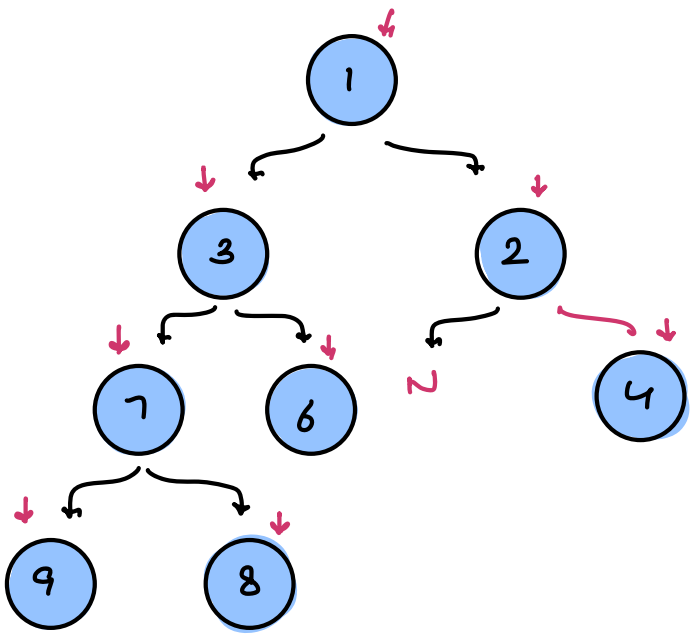
01. Invert Binary Tree
02. Equal tree partition
03. Next pointer BT
04. Root to leaf path sum = K
05. Diameter of BT (Amazon)

* Next pointer in perfect BT with SC : O(1)

* Invert the Binary Tree







⇒ swapping → preorder recursive calls

Idea → For all nodes, swap left & right

* Can we do it in preorder?

```
void invert (root){
    if (root==null) {return}

    invert (root.left);
    invert (root.right);

    Node temp = root.left
    root.left = root.right
    root.right = temp;
}
```

TC: O(n) SC: O(h)

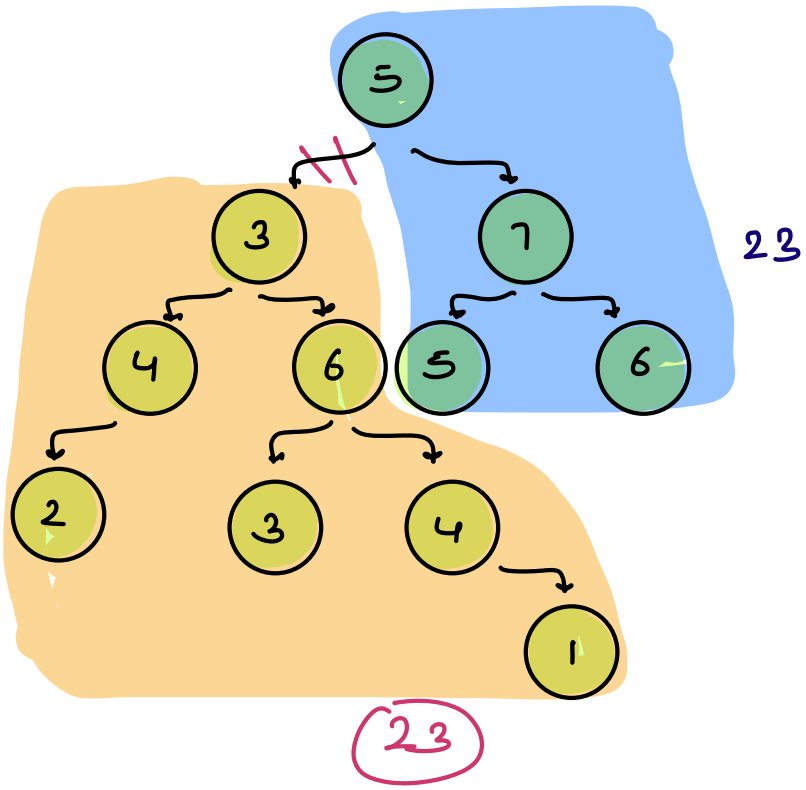
Asking lsub & rsub to go & invert themselves

swap

* Try if inorder is also possible?

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.



Ans = true

Obs - if sum of complete tree is odd, return false

if sum is even → check

Total sum = S, if sum of any subtree = $S/2$

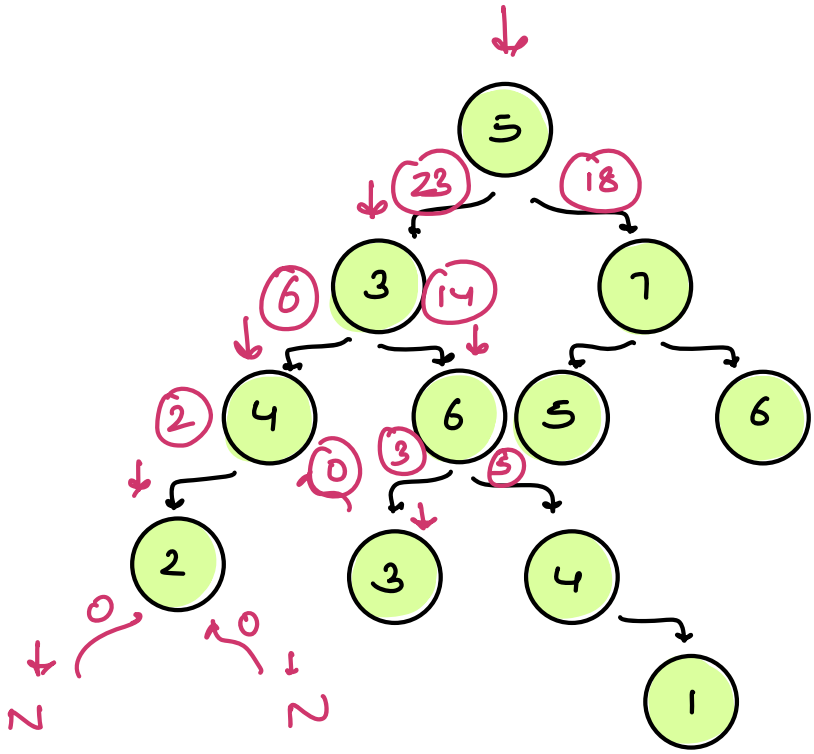
return true

```
public int sum(root)
    if (root == null) return 0;
    return sum(root.left) + sum(root.right) + root.val;
}
```

```
public int check(root)
    int s = sum(root);
    if (s%2 == 1) return false;
    find(root, s)
}
```

Travel & change

```
boolean ans = false
int find(root, s)
    if (root == null) return 0;
    l = find(root.left, s)
    r = find(root.right, s)
    if (l == s/2 || r == s/2) {
        ans = true;
    }
    return l + r + root.val;
}
```



* pair {

```
int sum;
boolean equal;
```

}

```
public pair fun ( root, s)
    if (root == null) return new pair(0, false)
    pair l = fun (root.left, s)
    pair r = fun (root.right, s)
    if ( l.equal == true || r.equal == true) {
        return new pair(0, true);
    }
```

Treenode x ;

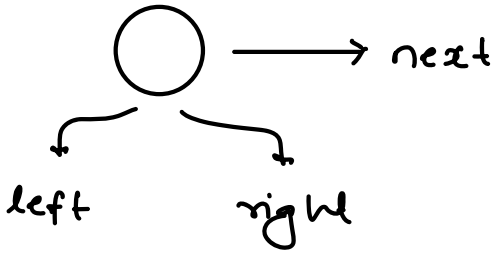
```
if(l.equal==true)
    if(x==null)
        x = root;
```

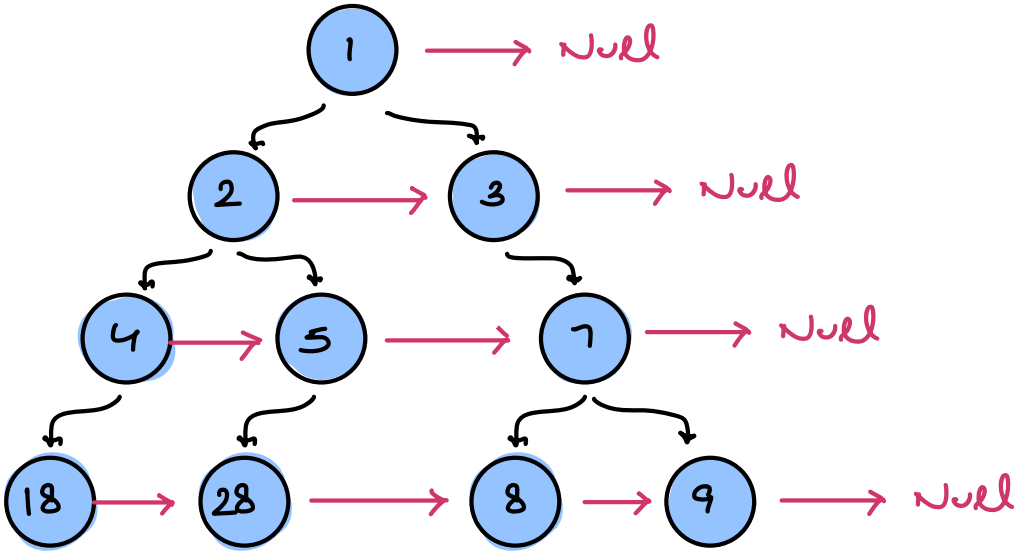
```
    else if ( l.sum == s/2  ||  r.sum == s/2 )
    |   return new pair (0, true);
    }
    else {
    |   return new pair ( l.sum + r.sum + root.val, false);
    }
}
```

Next pointer in Binary Tree

Initially for all nodes, next pointer points to NULL

Update next pointer to point to the next node in same level. (Left to Right)






```
Queue<Node> q ;

q.enqueue(root);

while (q.size()>0)
    int sz= q.size();

    for (i=1; i≤ sz ; i++){
        Node rem = q.dequeue();
```

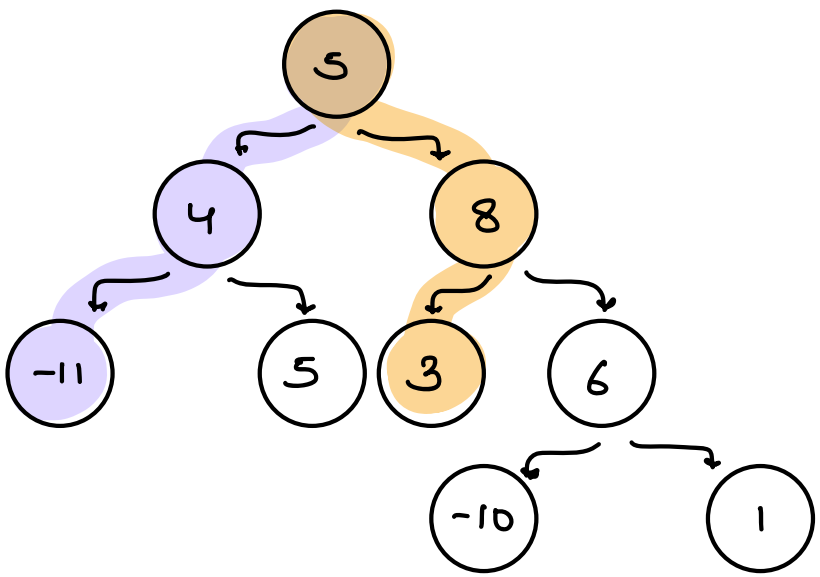
TC: O(n)

SC: O(n)

```
        if (i == sz) rem.next = null;
        else rem.next = q.front();
        print(rem.data);
        if (rem.left != null) q.enque(rem.left);
        if (rem.right != null) q.enque(rem.right);
    }
    println();
}
```

10:09 pm → 10:19 pm

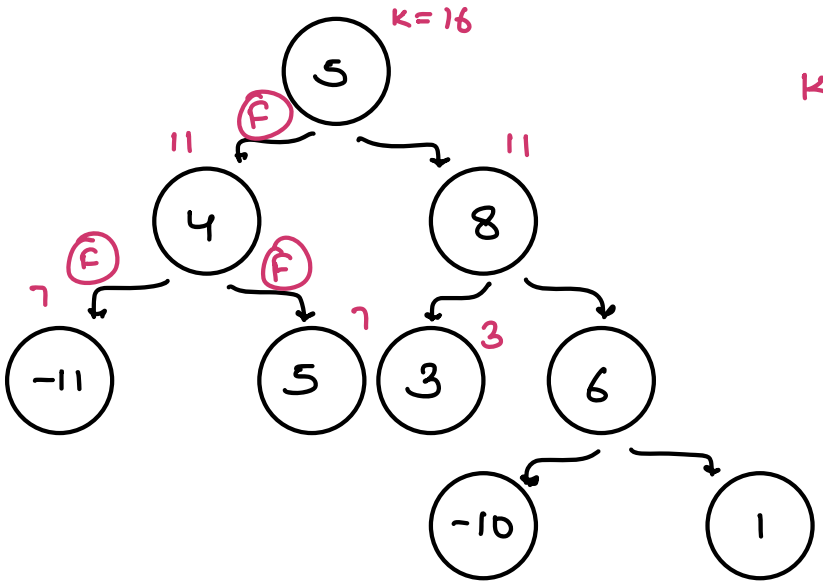
Q Check if root to leaf path sum equals to K



K = 16 → True

K = -2 → True

K = 10 → false



K=16

1

TC: O(N)

SC: O(H)

```
boolean check(root, k)
{
    if (root == null) return false

    if (root.left == null && root.right == null){

        return (root.val == k)

    }

    return check(root.left, k-root.val) ||

           check(root.right, k-root.val);

}
```

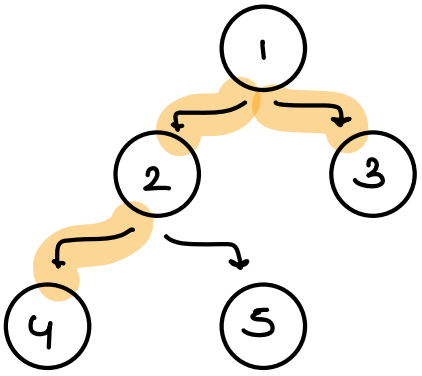
leetcode → Path Sum

Path Sum II

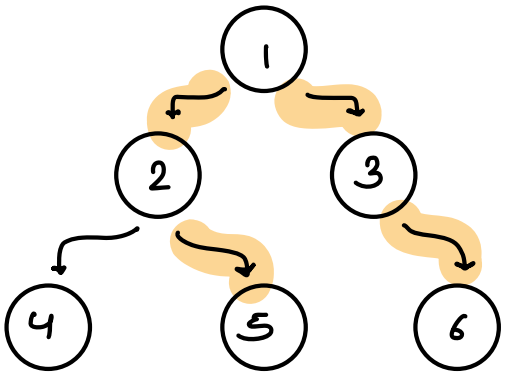
Path Sum III

Q Diameter of Binary Tree

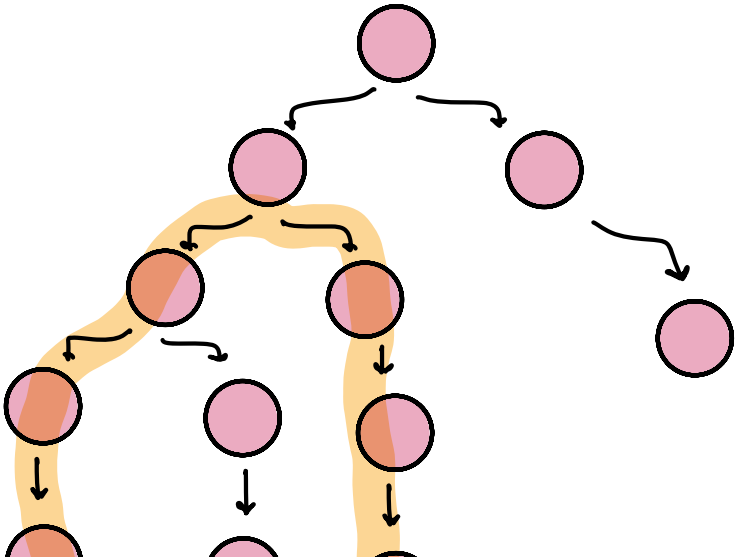
Diameter → It is no. of edges on largest path b/w two nodes in a binary Tree

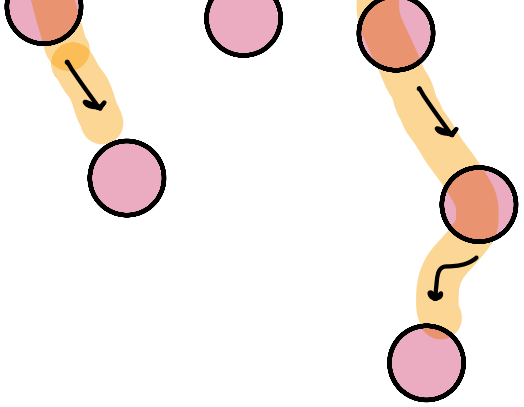


Ans = 3



Ans = 4





* 3 possibilities

→ Dia can lie on LHS

→ Dia can lie on RHS

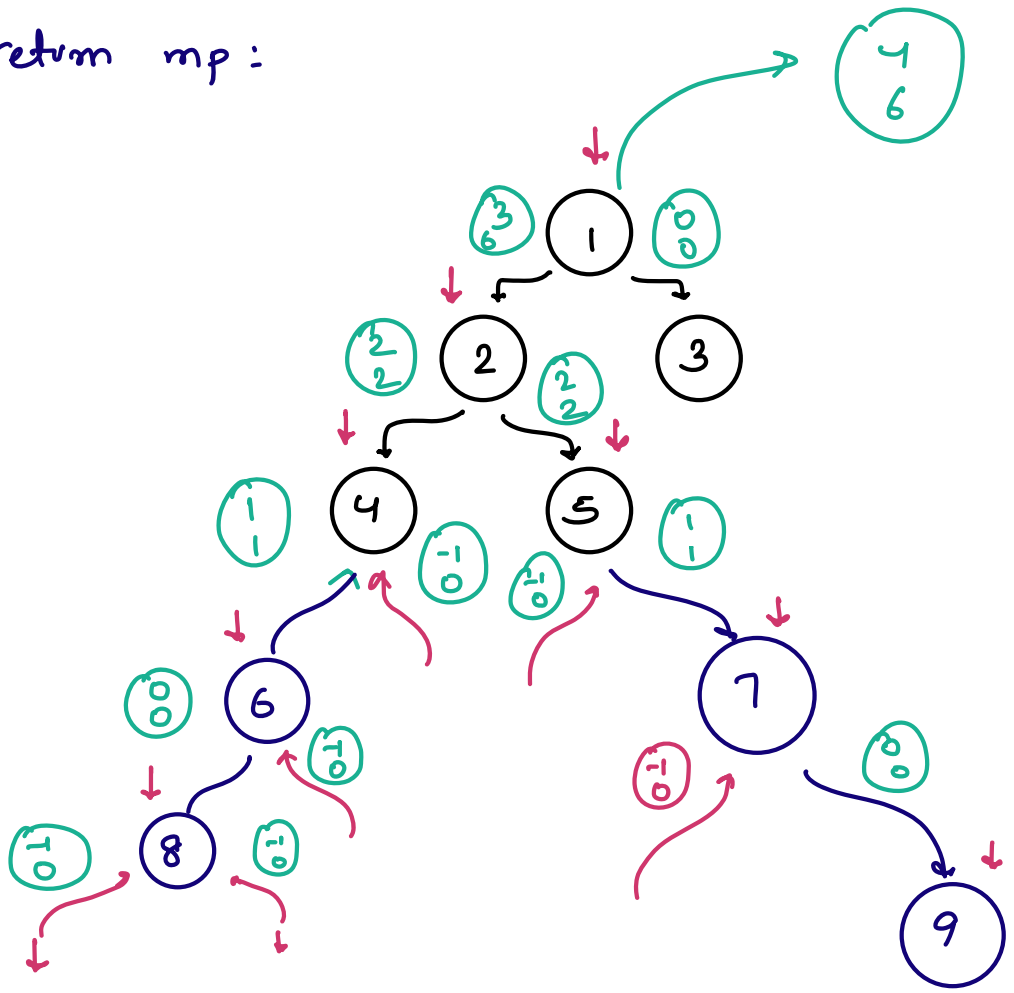
→ Dia is passing through root = lh + rh + 2

```
void main(){
    return diameter(root).dia;
}
```

* Public class Dpair{

```
    int ht;
    int dia;
}
```

```
public Dpair diameter(root)
{
    if (root == null) return new Dpair(-1, 0);

    Dpair l = diameter(root.left)
    Dpair r = diameter(root.right)

    Dpair mp = new pair();

    mp.ht = Math.max(l.ht, r.ht) + 1;

    mp.dia = Math.max(l.dia, r.dia, l.ht + r.ht + 2);

    return mp;
}
```

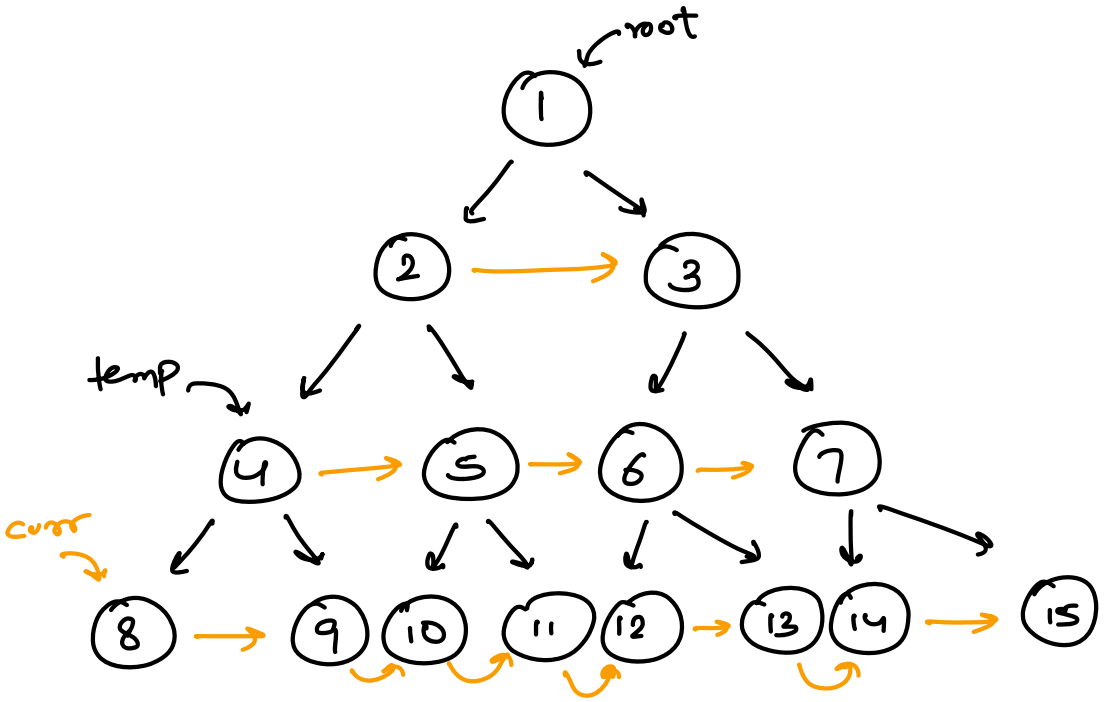


```
int dia = 0;
```
→ Global variable

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

    lh = height (root.left)

    rh = height (root.right)

    dia = Max ( dia, lh+rh+2);

    return Math.max (lh, rh)+1;
}
```

* Fill next pointer in Perfect Binary Tree

Expected SC : O(1)




```
Node curr = root

while (curr.left != null) {

    Node temp = curr;

    while (curr != null)
        curr.left.next = curr.right;
        if (curr.next != null) {
            curr.right.next = curr.next.left
        }
        curr = curr.next

    curr = temp.left
}
```

TC : O(n)

SC : O(1)