Agenda

i) Why trees ?

ii) Terms related to trees

iii) Structure of trees

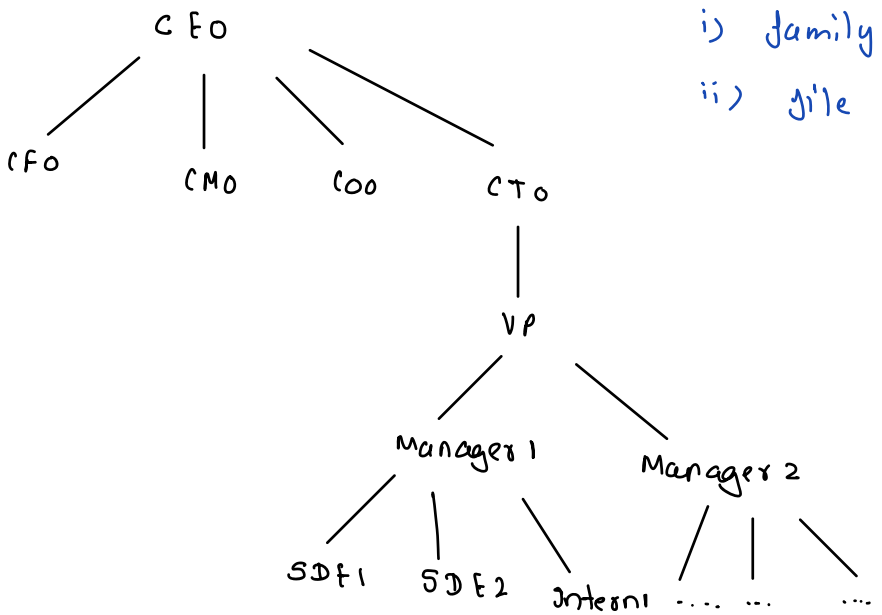
iv) Traversal (In, pre, post)

v) Questions
   → size, sum, height

Linear DS: Arrays, stacks, queue } → continous memory allocation

LL } → discontinous memery allocation

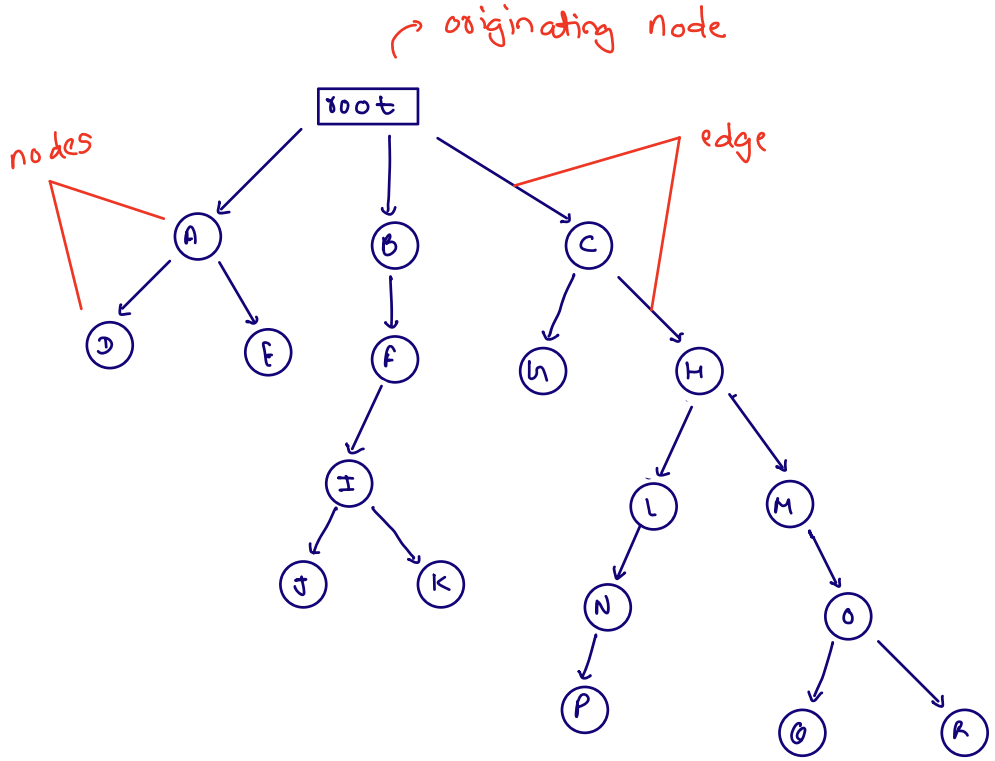
Non Linear DS / hierarchical : Trees



i) family tree

ii) file manager

Important terms



1) **Parent - child**: every node has a single parent only

2) **Siblings**: nodes with same parent are known as siblings.

3) **Leaf node**: nodes with 0 child

4) **Ancestor**: nodes coming in the path of root to this node.

5) **Descendant**: all nodes coming under this node.

quick question:

i) Parent of M → H

ii) Are N & O siblings ? No

iii) Is N a leaf node ? No

iv) Ancestors of L ? root, C, H

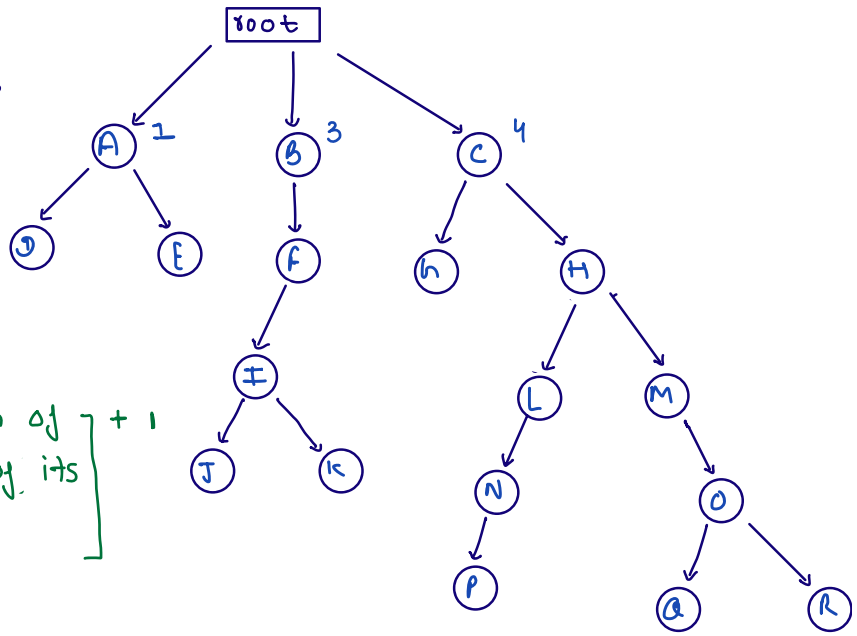
v) Descendants of M ? O, Q, R

1) **height (node)**

The max distance b/w this node to any of its descendant leaf node.

height (leaf node) = 0

height (node) = [maximum of heights of its child] + 1

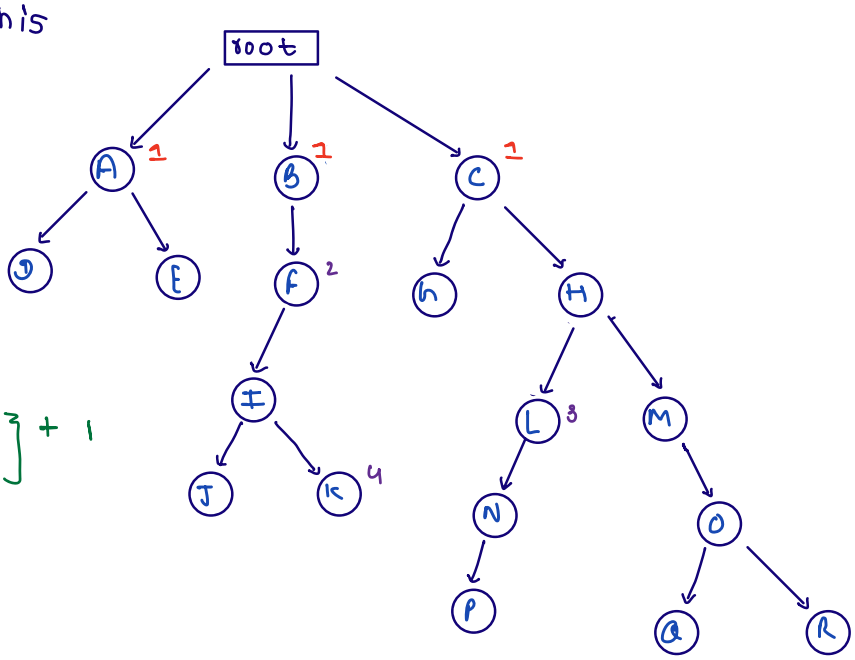


2) **Depth (node)**

distance b/w root and this node.

depth (root) = 0

depth (node) = [depth of its parent] + 1

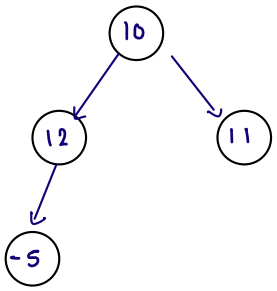


Structure of trees

N-ary (generic tree)

Binary tree : Every node can have atmax 2 childs.

↳ 0, 1, 2

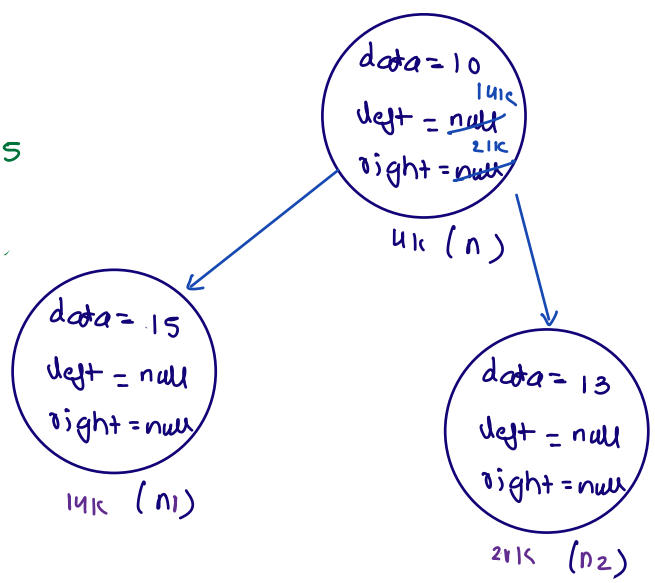



```
class Node {
    int data;
    Node left;
    Node right;
    Node (int data) {
        this.data = data;
    }
}
```

```
Node n = new Node (10);
Node n1 = new Node (15);
Node n2 = new Node (13);
n.left = n1;
n.right = n2;
```

```
sopln (n.left.data); → 15
sopln (n.left.left.data);
```

null ptr exception

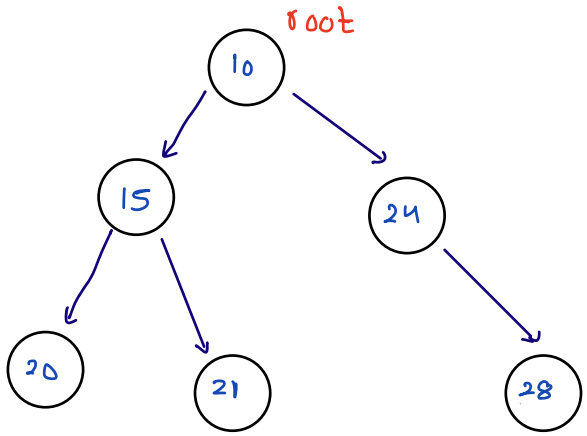


note: Don't worry about creation of tree, root node of already constructed will be given to us in ques

Traversal of binary tree

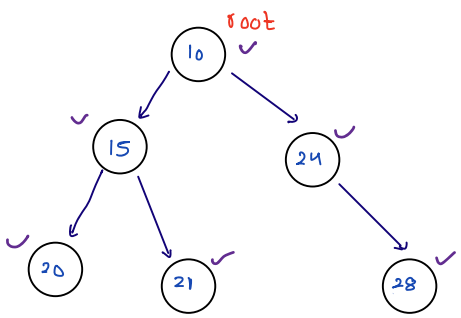
i) iterative (tricky) {upcoming classes}

ii) recursive (easy)




```
void traversal (Node root) {
    if (root == null) {
        return;
    }
    traversal (root.left);
    traversal (root.right);
}
```

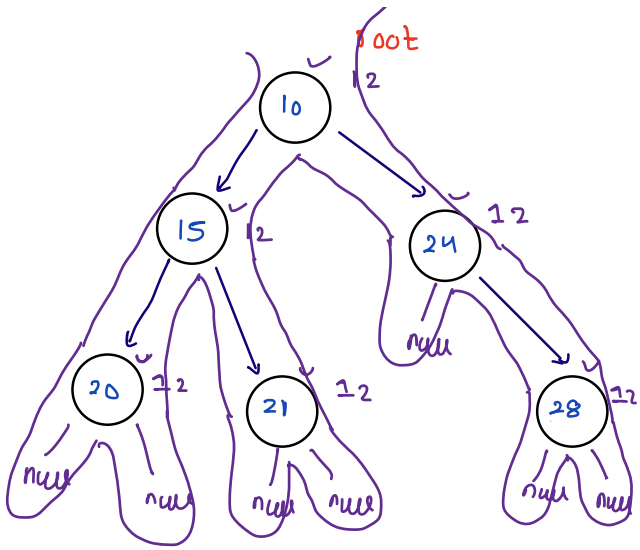
```
void traversal (Node root) {
    if (root == null) {
        return;
    }
    1. traversal (root.left);
    2. traversal (root.right);
}
```



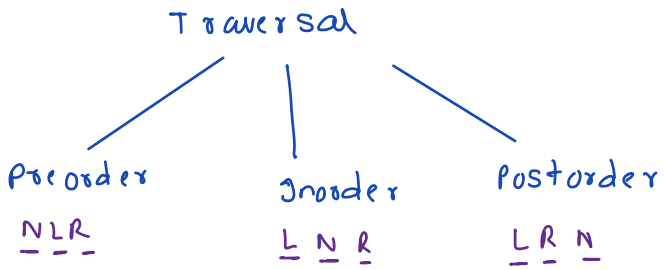
travel	~~root = null~~	
travel	~~root = null~~	
travel	~~root = 28~~	1 2
travel	~~root = null~~	
travel	~~root = 24~~	1 2
travel	~~root = null~~	
travel	~~root = null~~	
travel	~~root = 21~~	1 2
travel	~~root = null~~	base
travel	~~root = null~~	base
travel	~~root = 20~~	1 2
travel	~~root = 15~~	1 2
travel	~~root = 10~~	1 2

```
void traversal (Node root) {
    if (root == null) {
        return;
    }
    1. traversal (root.left);
    2. traversal (root.right);
}
```

TC: O(n)



Pre, In, Post




```
void traversal (Node root) {
    if (root == null) {
        return;
    }
    sopln (root.data);
    traversal (root.left);
    traversal (root.right);
}
```

→ preorder

```
void traversal (Node root) {
    if (root == null) {
        return;
    }
    traversal (root.left);
    sopln (root.data);
    traversal (root.right);
}
```

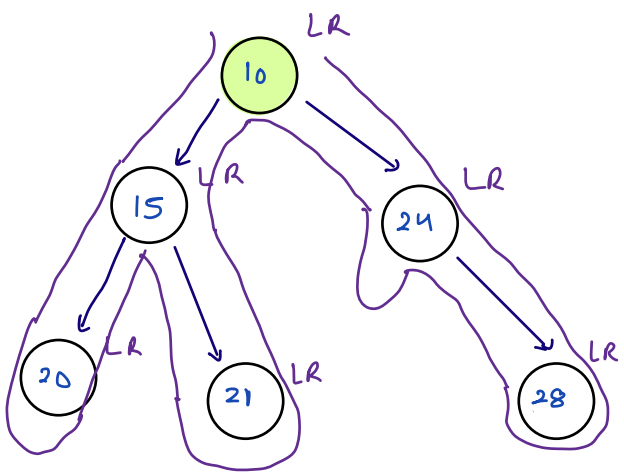
→ Inorder

```
void traversal (Node root) {
    if (root == null) {
        return;
    }
    traversal (root.left);
    traversal (root.right);
    sopln (root.data);
}
```

Postorder

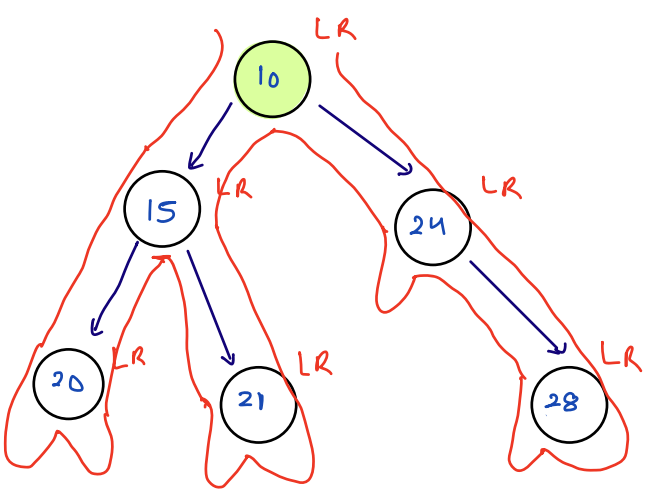
Dry run

```
void traversal (Node root) {
    if (root == null) {
        return;
    }
    sopln (root.data);
    traversal (root.left);
    traversal (root.right);
}
```



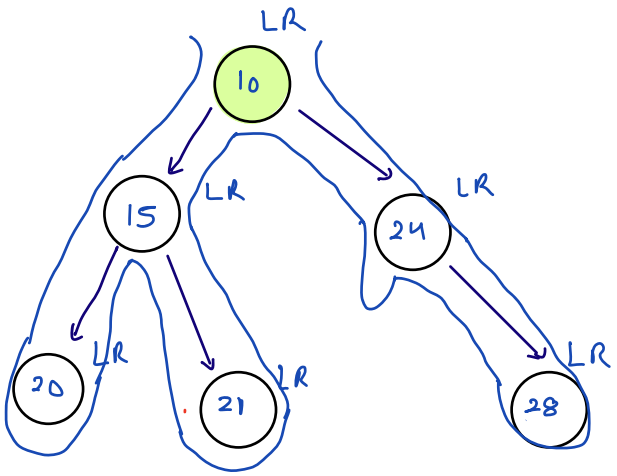
Preorder : 10 15 20 21 24 28

```
void traversal (Node root) {
    if (root == null) {
        return;
    }
    traversal (root.left);
    sopln (root.data);
    traversal (root.right);
}
```



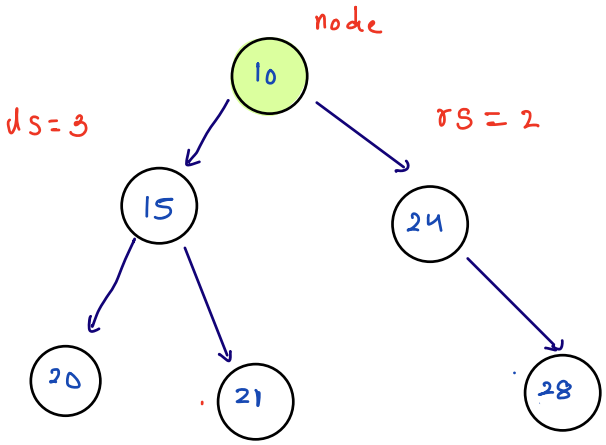
Inorder : 20 15 21 10 24 28

```
void traversal (Node root) {
    if (root == null) {
        return;
    }
    traversal (root.left);
    traversal (root.right);
    sopln (root.data);
}
```



Postorder : 20 21 15 28 24 10

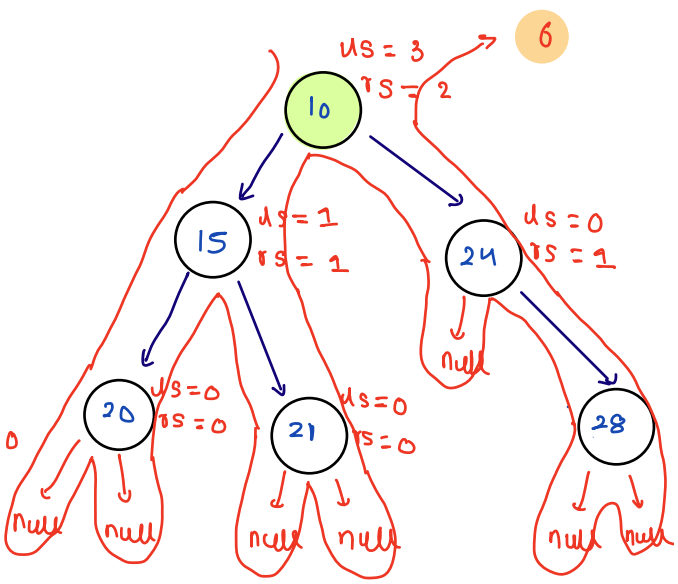
Q.1 Given root of a binary find its size (count nodes)



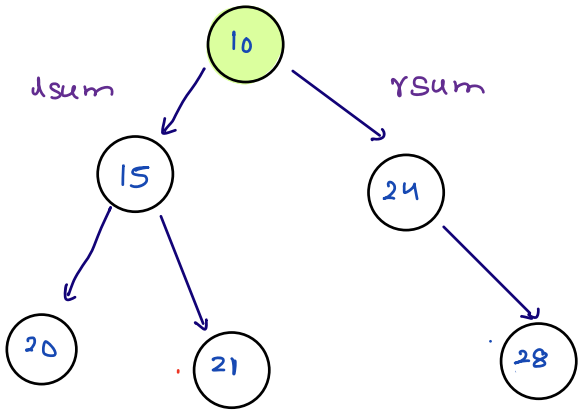
size = 6

ans = ls + rs + 1

```
int size(Node node) {
    if(node == null) {
        return 0;
    }

    int ls = size(node.left);
    int rs = size(node.right);
    return ls + rs + 1;
}
```



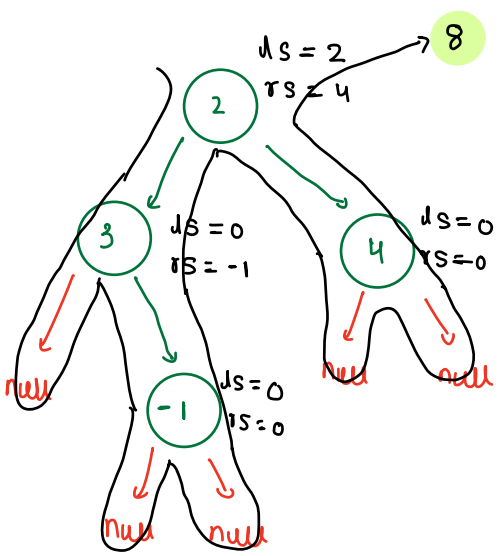
Q.1 Given root of a binary find its sum (sum of all nodes)



sum = 10 + 15 + 20 + 21 + 24 + 28

ans = lsum + rsum + node.data

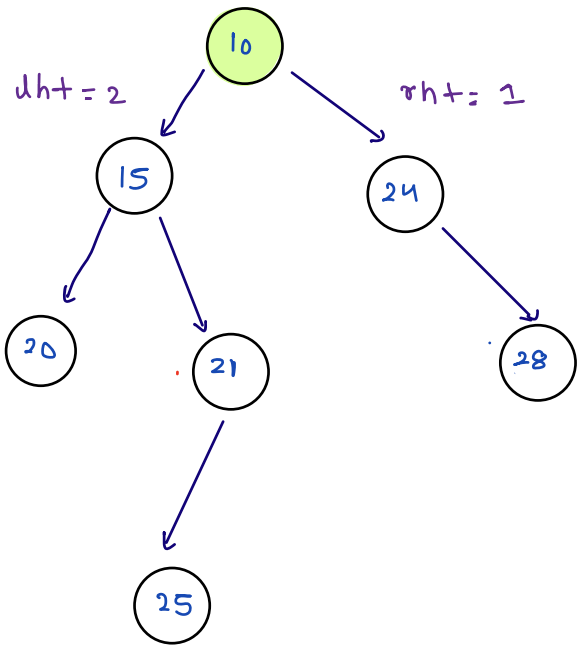
```
int sum(Node node) {
    if(node == null) {
        return 0;
    }

    int ls= sum(node.left);

    int rs= sum(node.right);

    return ls +rst node.data;
}
```



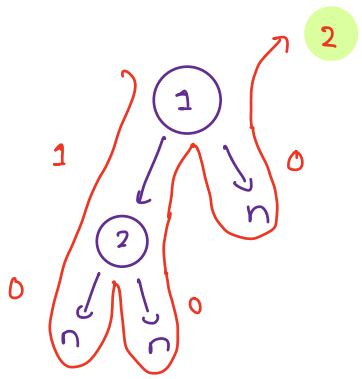
Q.3 Given root of a binary find height of tree.

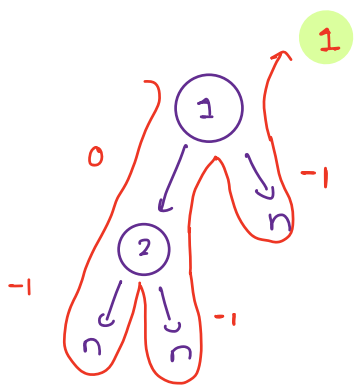
↳ (edge based)

height tree = height (root)



ans = 3

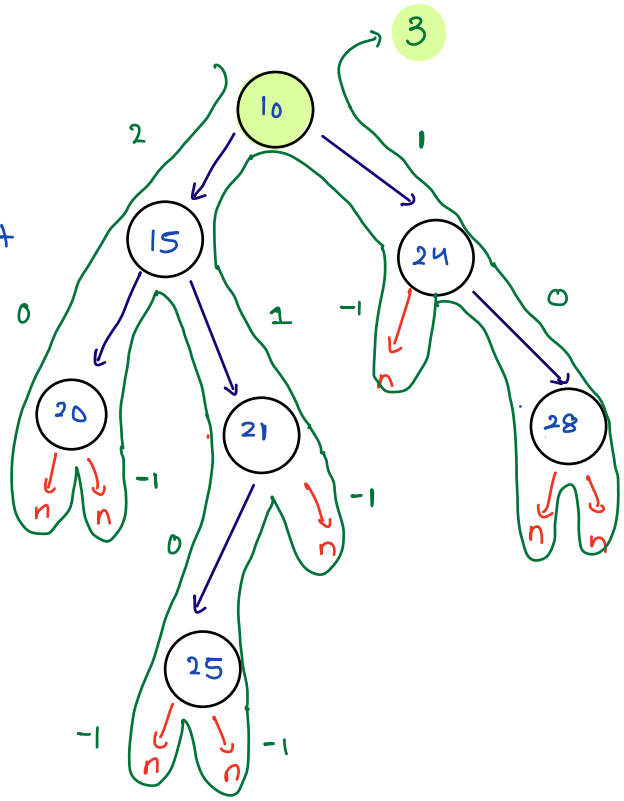




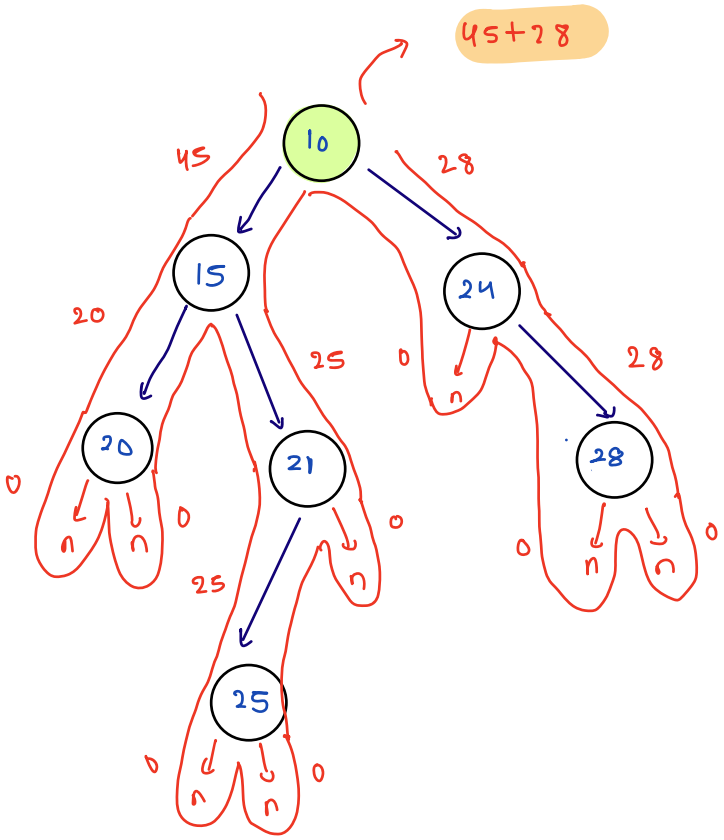

```
int height (Node node) {
    if (node == null) {
        return -1;
    }
    int lht = height (node.left);
    int rht = height (node.right);
    return Math.max(lht, rht) + 1;
}
```

Dry run

```
int height (Node node) {
    if (node == null) {
        return -1;        (node based ht
                          return 0)
                ↳ edge based ht
    }
    int lht = height (node.left);
    int rht = height (node.right);

    return Math.max(lht, rht) + 1;
}
```



Doubts




```
if( node == null ) {
        return 0;
}

la = call to left

ra = call to right

if (node.left == null &&
    node.right == null ) {
    return node.data;
}
else {
    return la+ra;
}
```