Trees

Rules

1) Attend the whole session (2 hrs) → 2:15 mins 20

2) Interview Problems not to be missed

Linked List is ? ✓

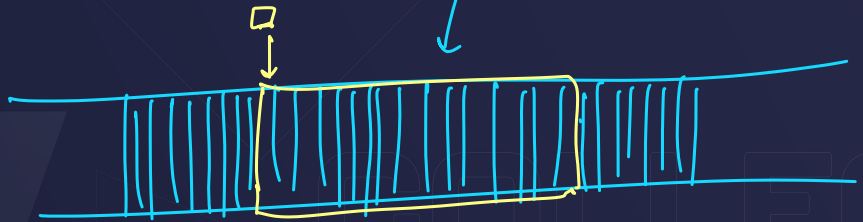
↳ Node

User defined data type

Arrays, List, LL, String

↓

linear data structures

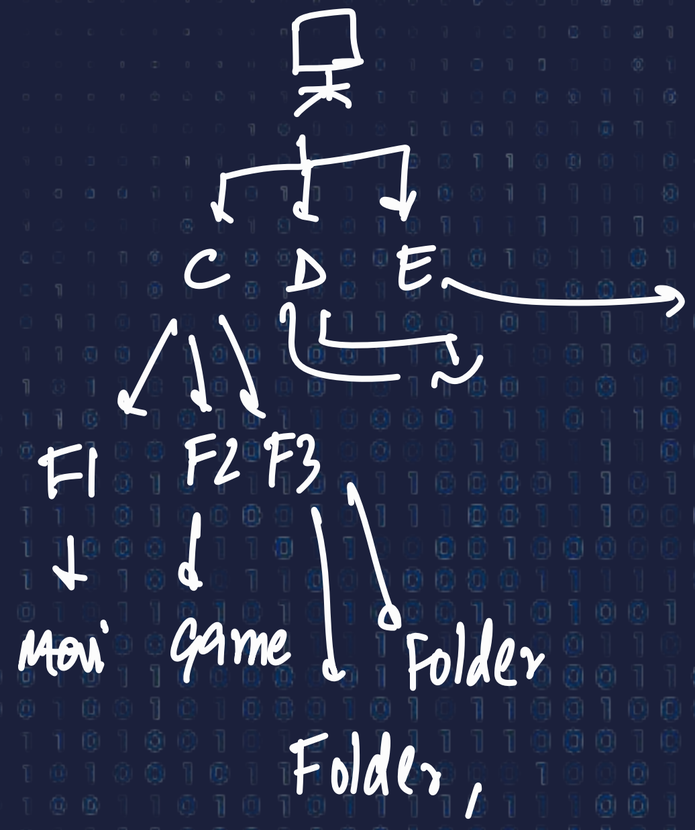


get → O(1) T.C.

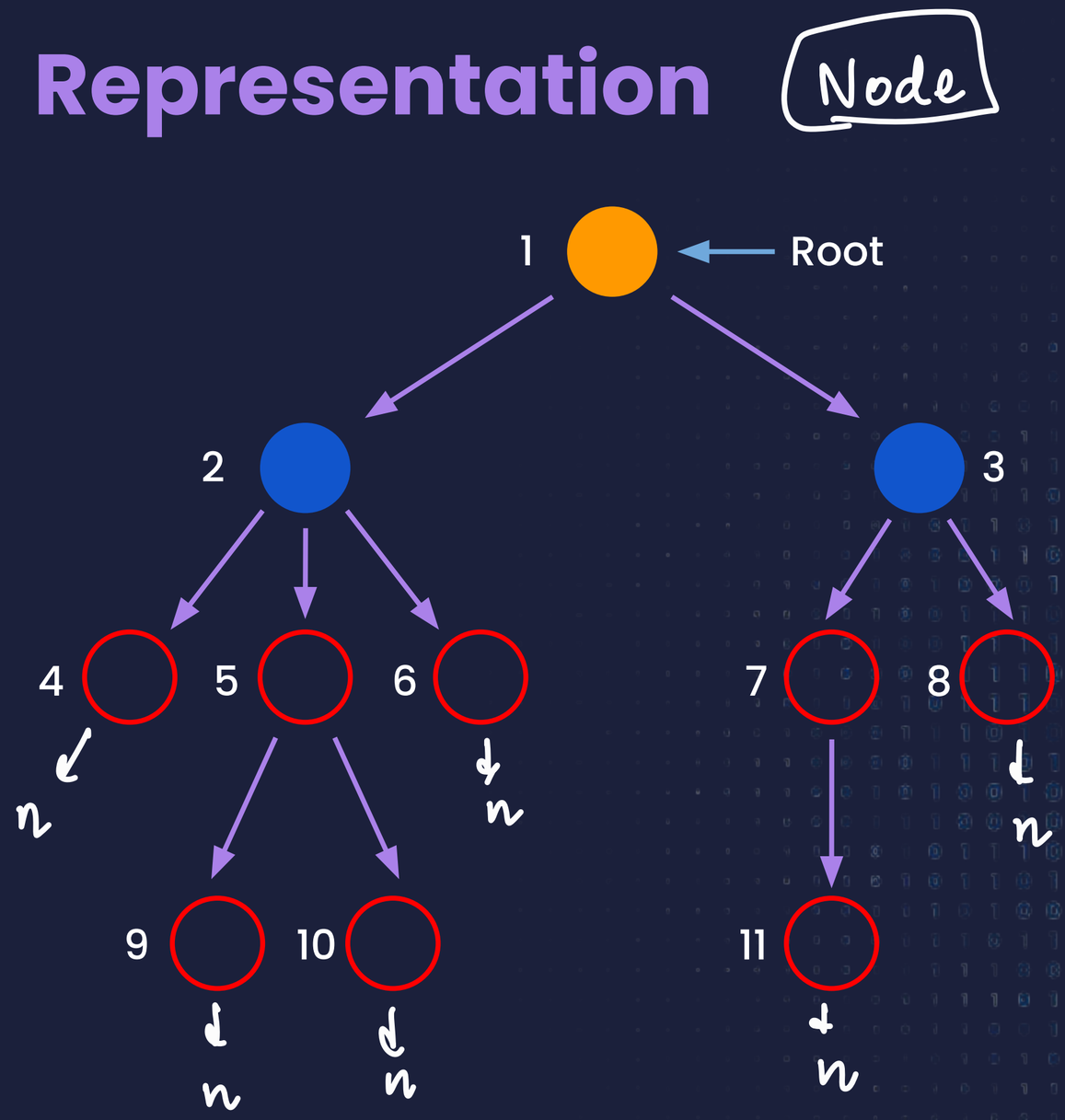
What is a Tree data structure

1) Non- Linear D.S.

2) Heirarchical D.S



Representation

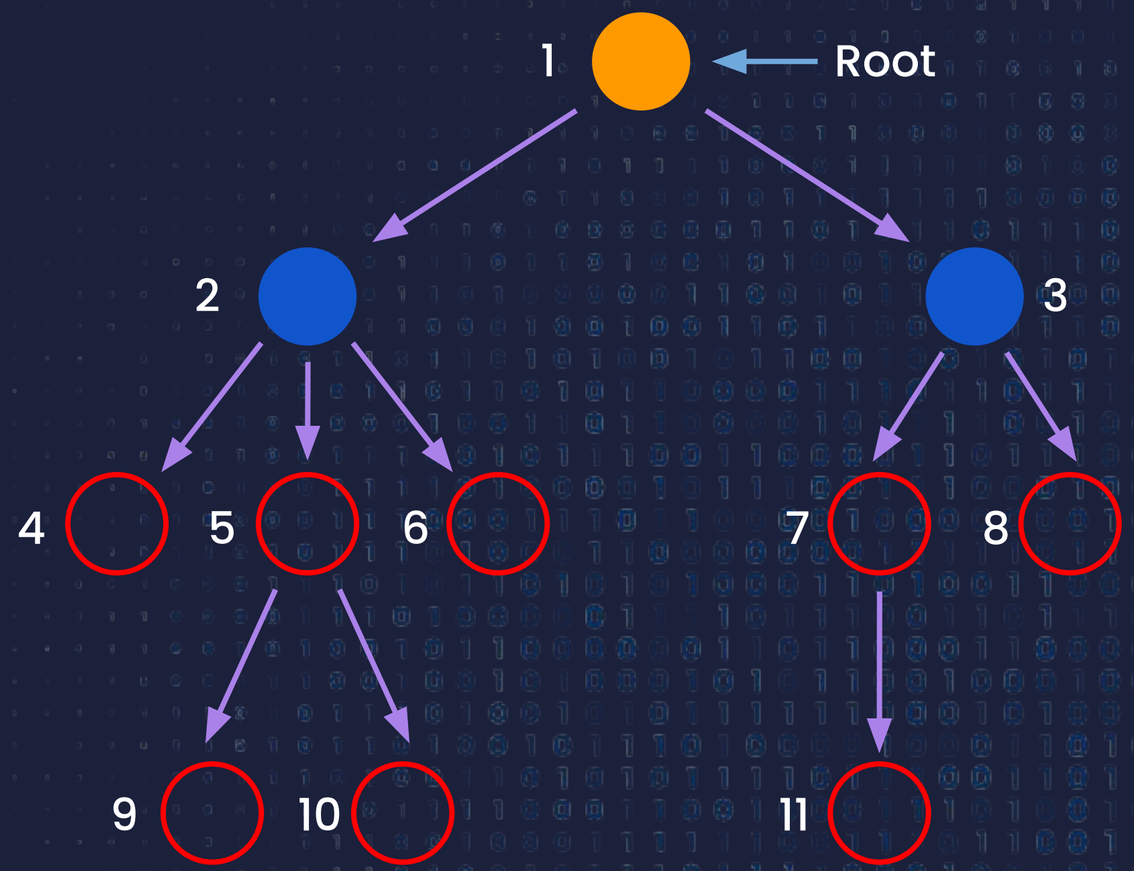


Terminology

1 child node cannot have 2 parent nodes

PW SKILLS

1. Root
2. Child Node
3. Parent Node
4. Sibling Nodes



1 → 2, 3

2 → 4, 5, 6

3 → 7, 8

4 → –

5 → 9, 10

6 → –

7 → 11

8 → –

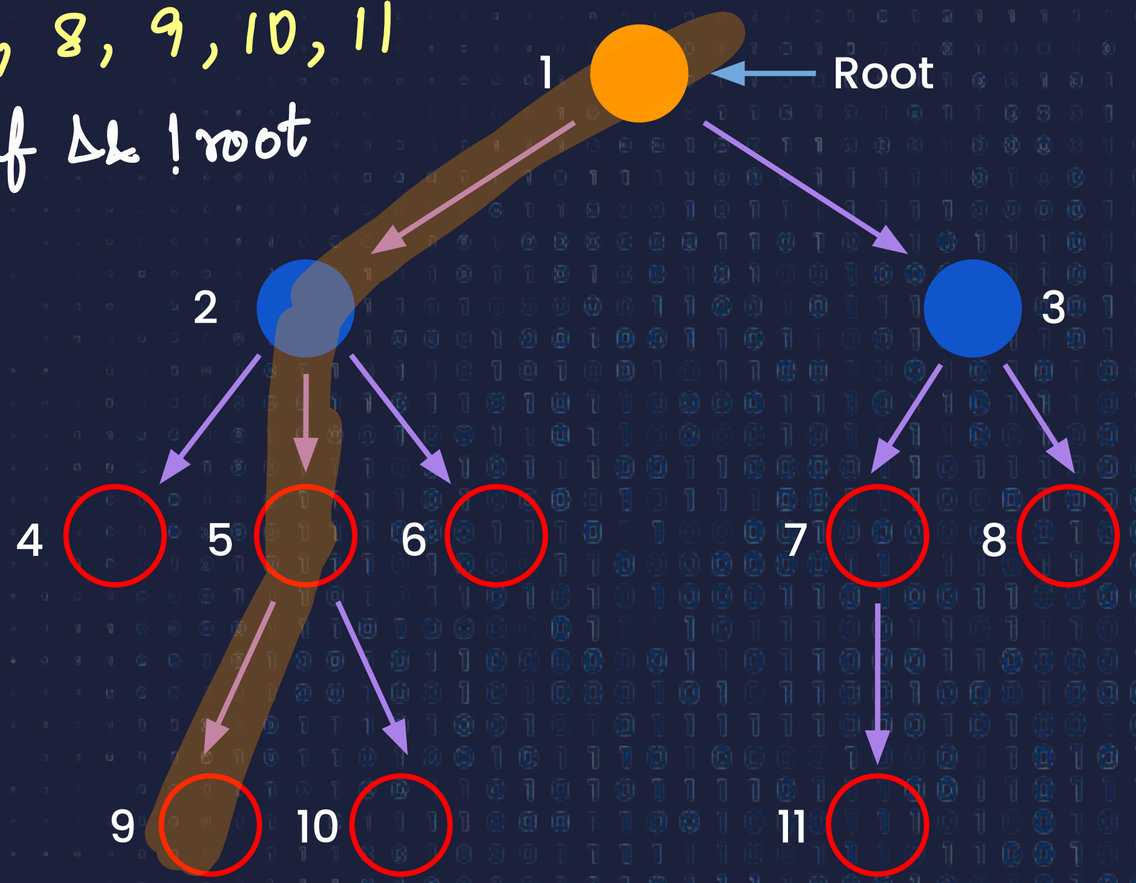
9 → –

10 → –

11 → –

Terminology

5. Leaf Node → 4, 6, 8, 9, 10, 11
6. Internal Node → !leaf && !root
7. Ancestor Node
8. Descendant Node



1 → 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

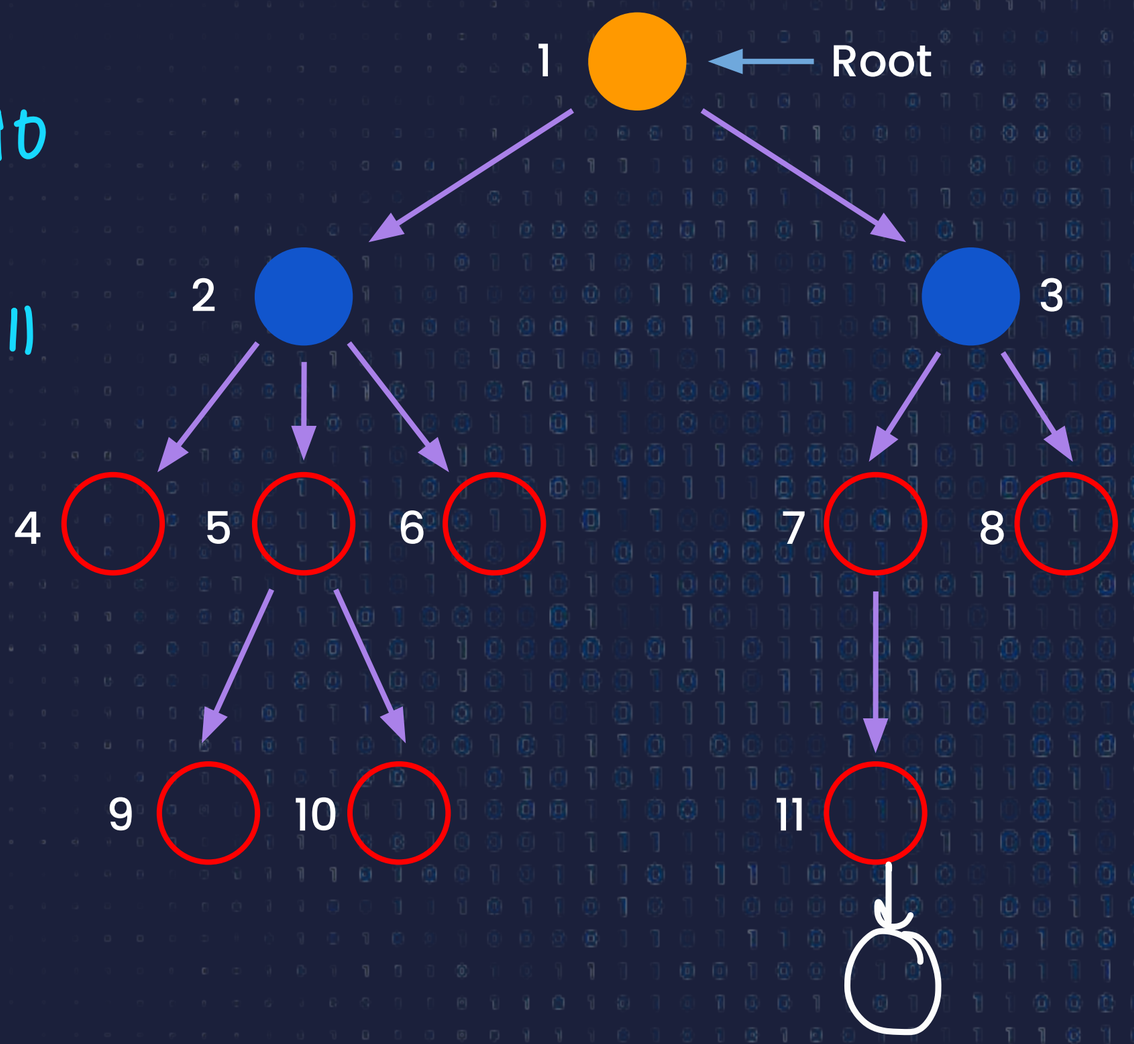
2 → 4, 5, 6, 9, 10

3 → 7, 8, 11

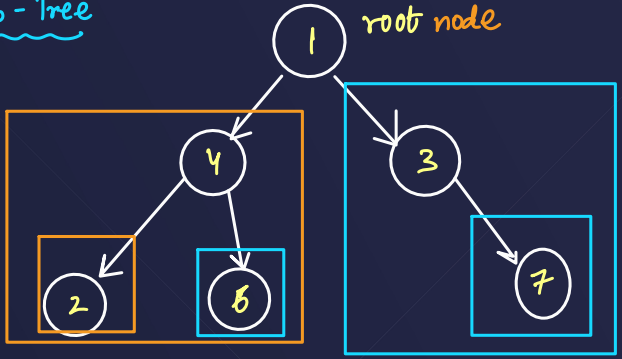
Terminology

Edges = Size - 1

9. Level → 4
10. Number of edges → 10
11. Height = level - 1
12. Size → no. of nodes = 11



Sub - Tree



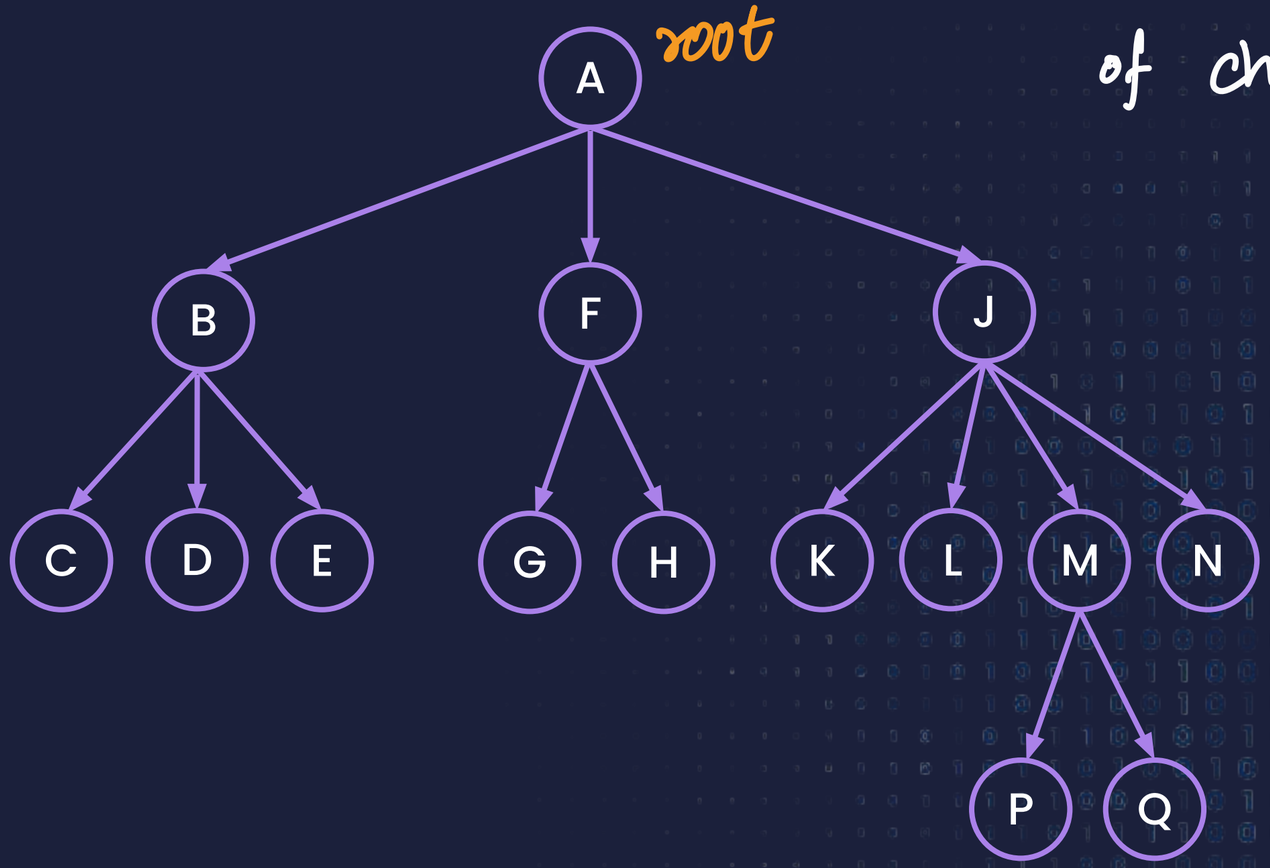
1) Left sub tree

2) Right sub tree

Important Properties of trees

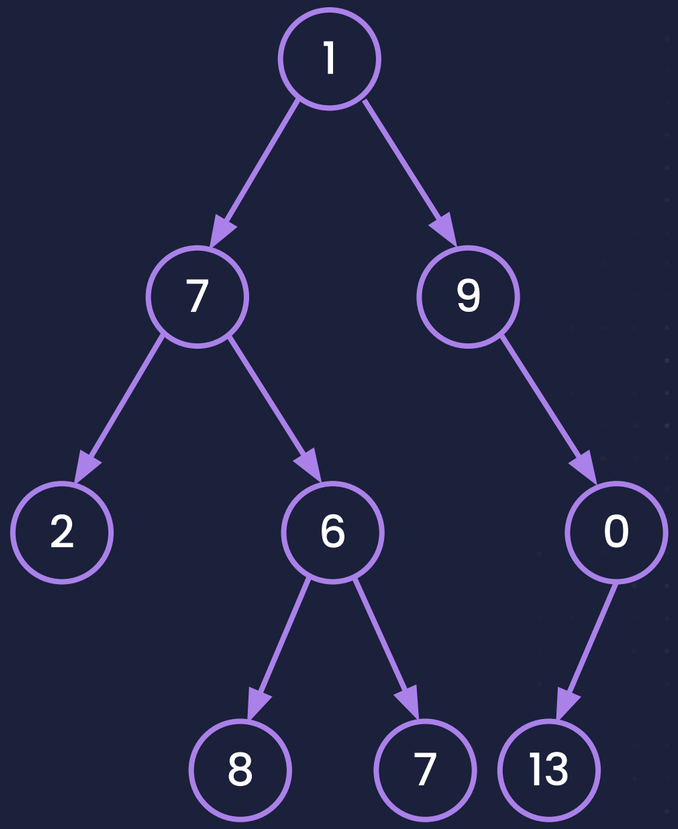
1. Traversing in a tree is done by depth first search and breadth first search algorithm.
2. It has no loop and no circuit.
3. It has no self-loop.

1. Generic Trees



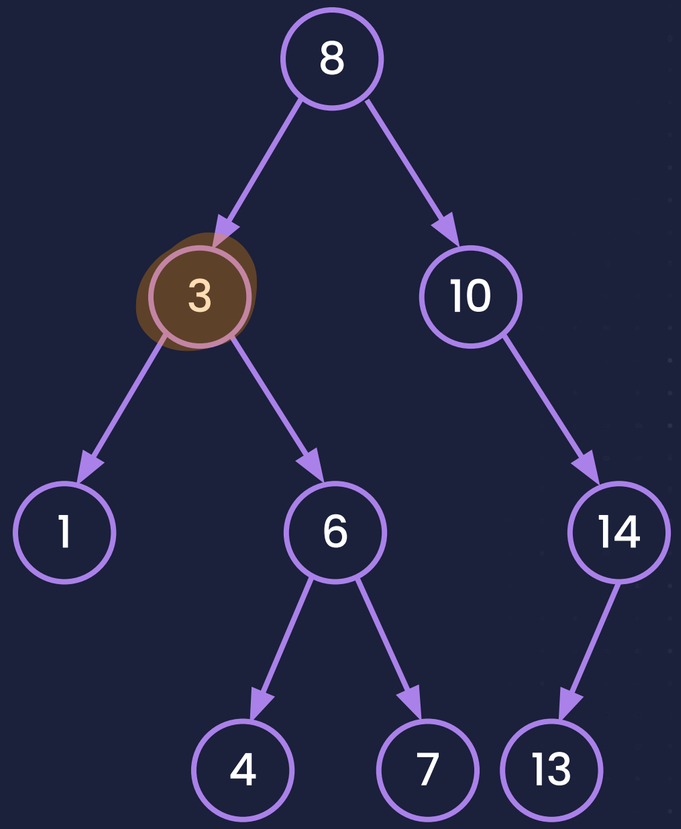
Each node can have any number of child nodes

2. Binary Trees



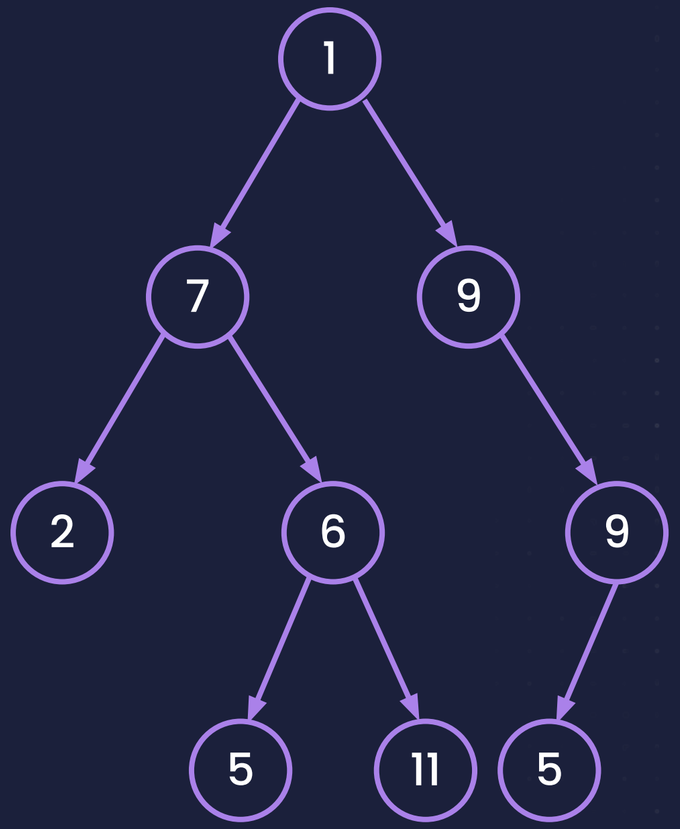
- Each node can have atmost 2 child nodes. → which are known as left child node & right child.

3. Binary Search Trees



- Each node can have atmost 2 child nodes. → which are known as left child node & right child.
- Every node to the left of a node is smaller & every node to the right has a greater value.

What are Binary Trees?

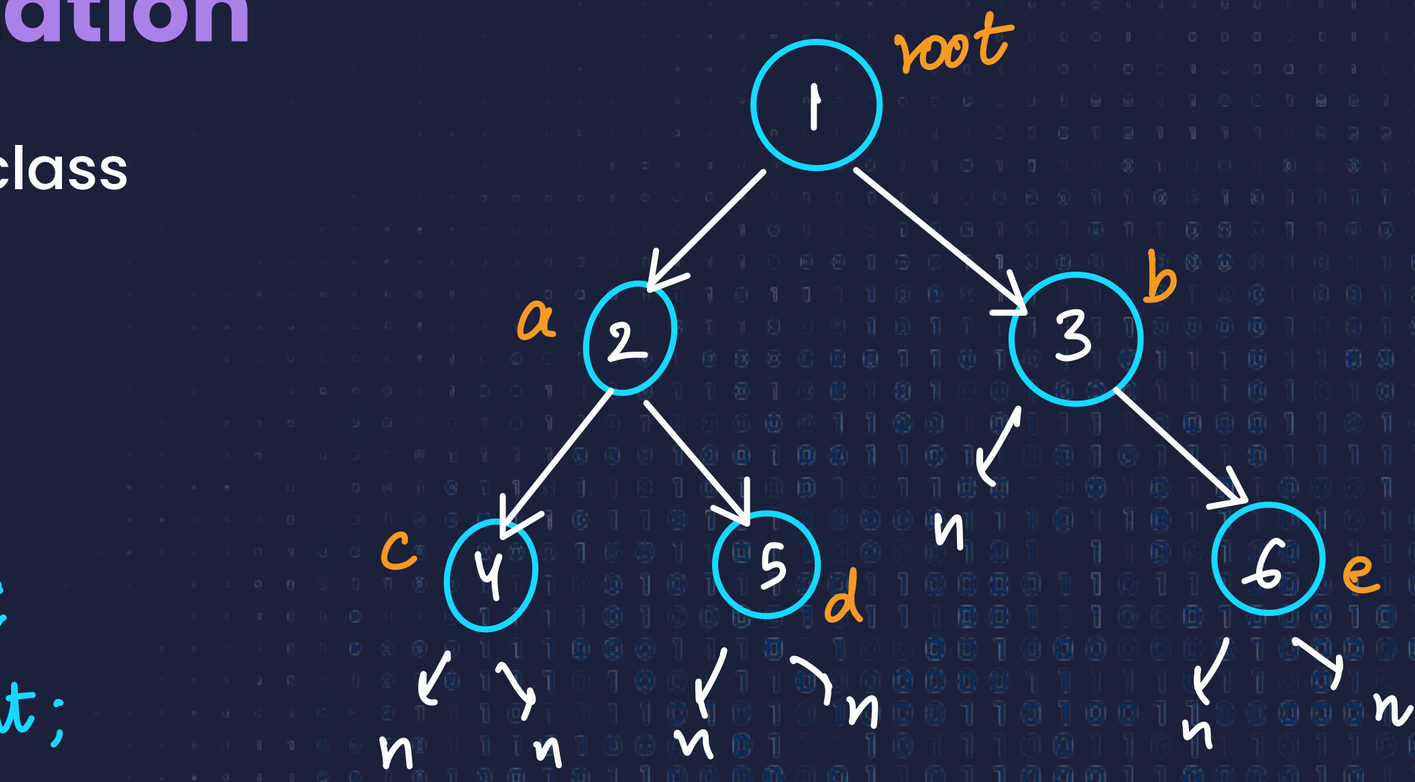


- Each node can have atmost 2 child nodes. → which are known as left child node & right child.

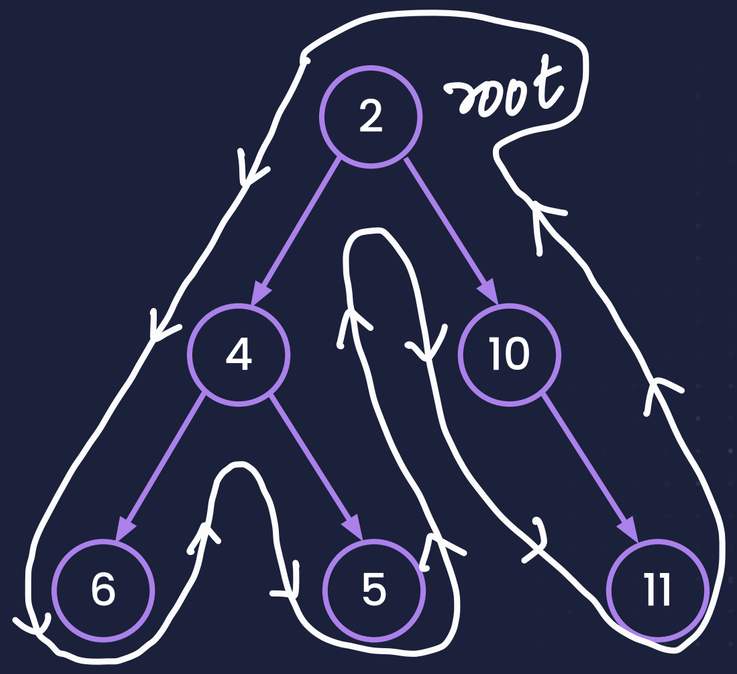
Implementation

Creating a Node class

```
class Node {
    int val;
    Node left;
    Node Right;
}
```



Display →



Recursion → root

2 → 4, 10

4 → 6, 5

10 → n, 11

6 → n, n

5 → n, n

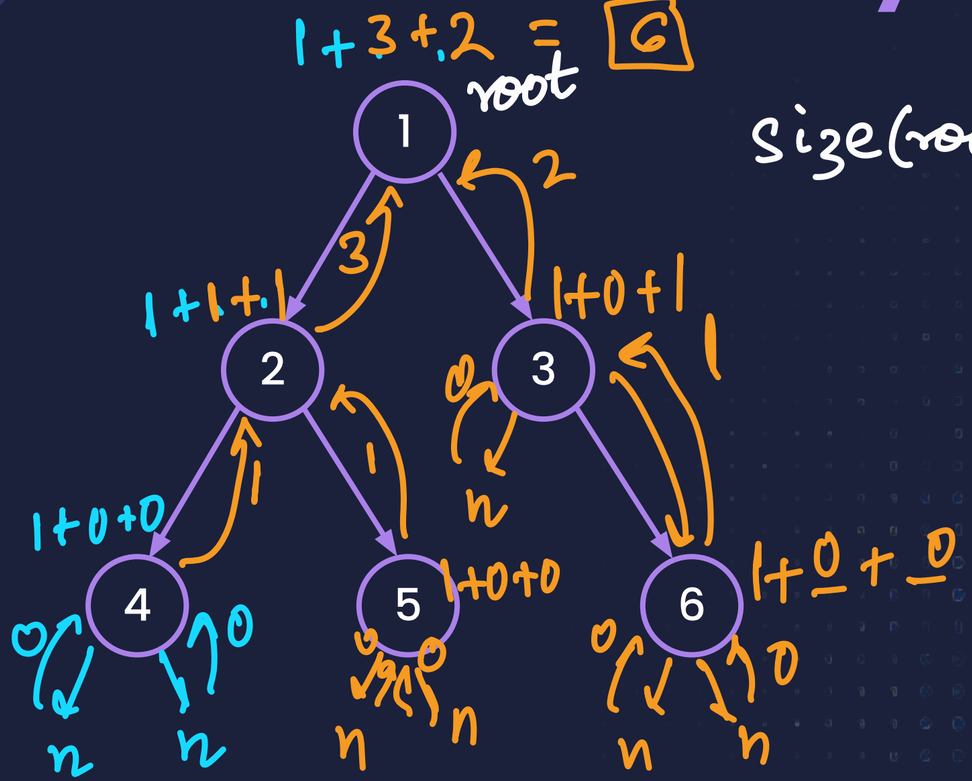
11 → n, n

```
2 -> 4 10
4 -> 6 5
6 ->
5 ->
10 -> 11
11 ->
```

Preorder

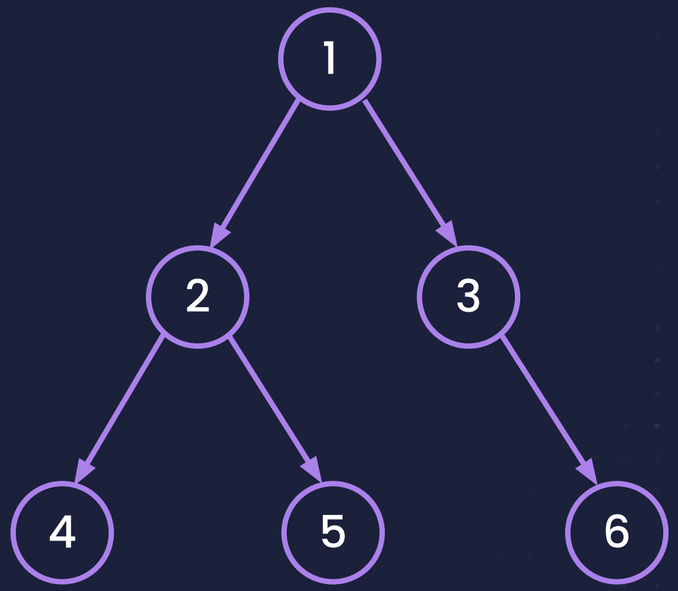
Find size of Binary Tree

= no. elements present



$$Size(root) = 1 + Size(root.left) + Size(root.right)$$

Find size of Binary Tree



size(1) = 1 + size(2) + size(3)

size(2) = 1 + size(4) + size(5)

size(4) = 1 + size(n) + size(n)

1 1 0 0

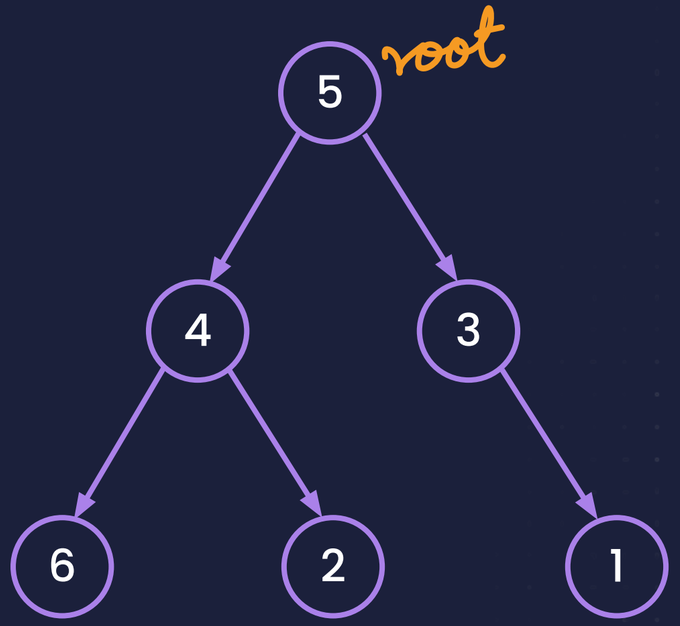
```
int size(Node root){

    return = 1 + LeftSize + RightSize;

}
```

3

Find sum of tree nodes



$Sum(root) = root.val + sum(root.left) + sum(root.right)$

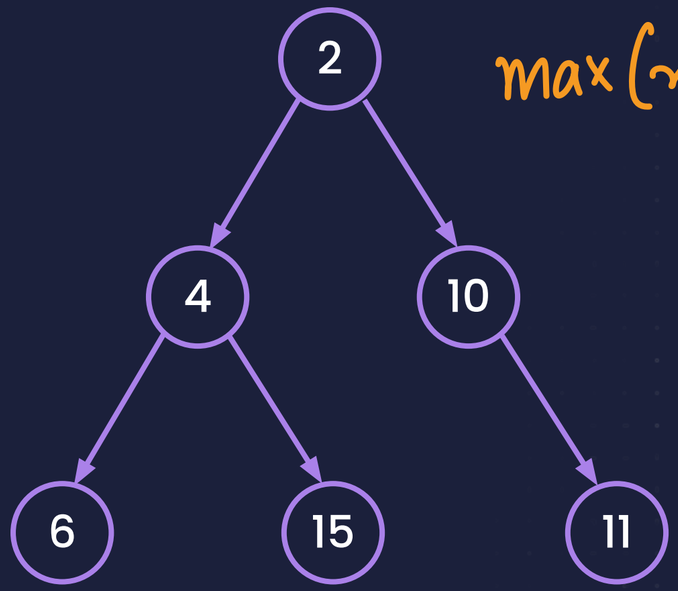
T.C. = $O(n)$ where 'n' is no. of nodes

& if n = no. of levels then

T.C = $O(2^n)$

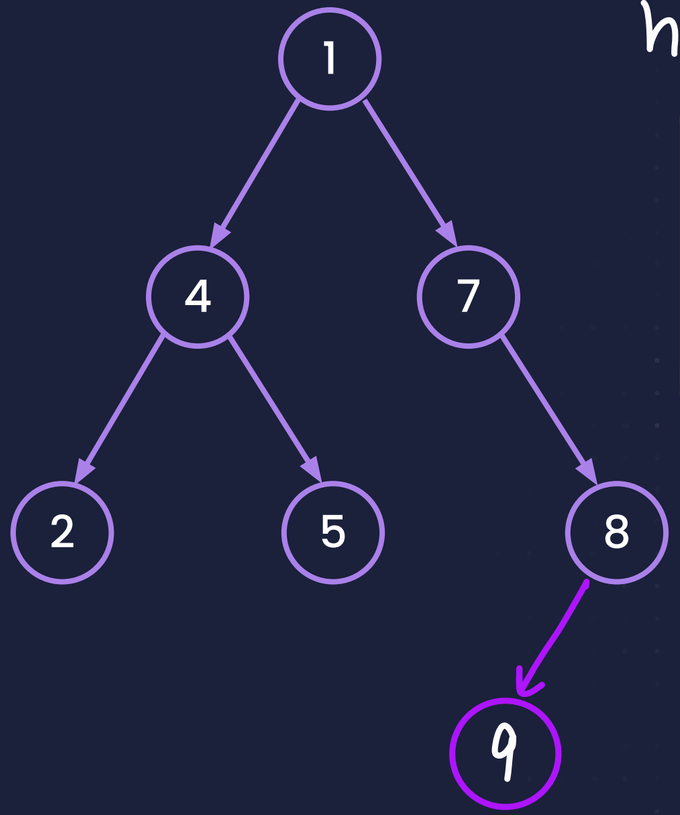
$5 + 4 + 3 + 6 + 2 + 1 = \frac{6 \cdot 7}{2} = \boxed{21}$

Find node with max value



max(root) = max [root.val , max(root.left), max (root.right)]

Find height of Binary Tree



height(root) = 1 + max(height(root.left), height(root.right))

```
public static int height(Node root){
    if(root==null) return 0;
    return 1 + Math.max(height(root.left),height(root.right));
}
```

leaf node → return 0;

Size, Sum, Height, MaxValue

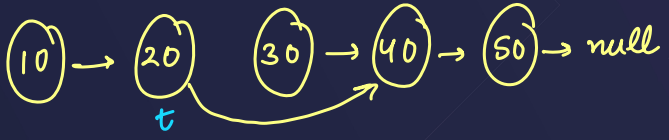
↓

Root & Left Subtree & RightSubtree

Homework:

→ minValue in the tree

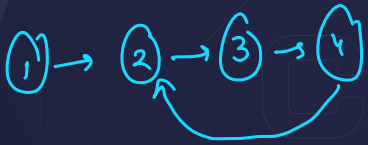
→ product of the tree.

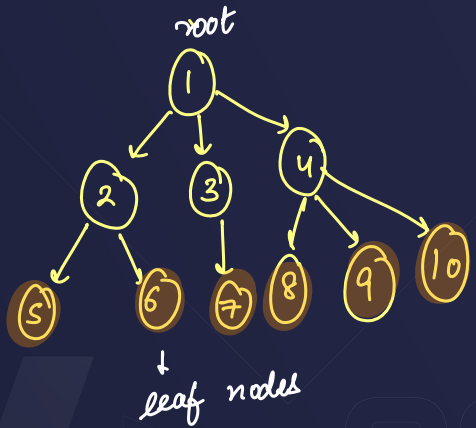


Trees

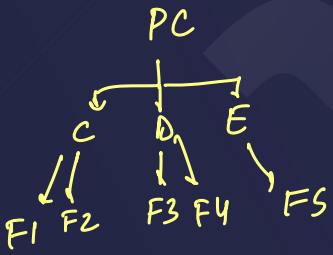
```
class Node{
    int val;
    Node next;
}
```

t.next = t.next.next





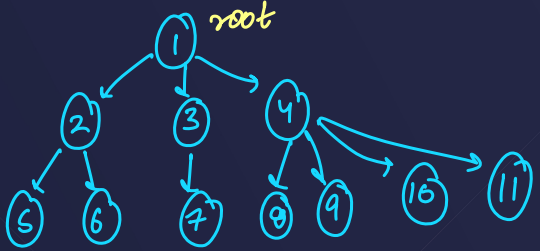
Ancestors & Decendants

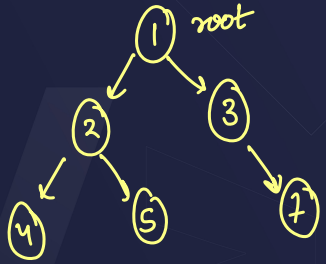


Heirarchy

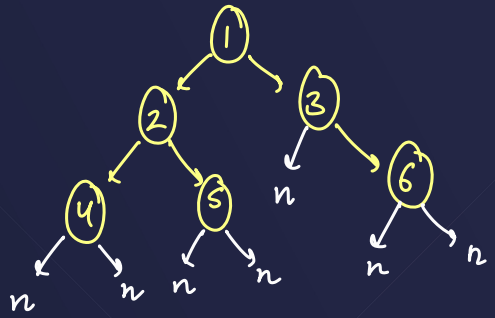
Upside down

Generic Tree

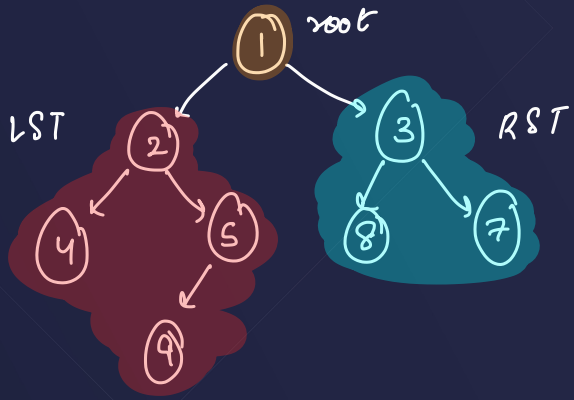


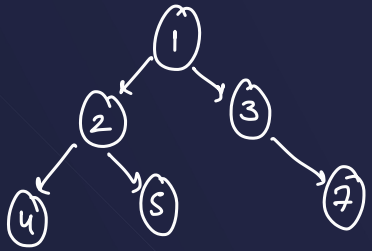


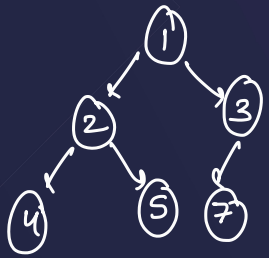

```
class Node {
    int val;
    Node left;
    Node right;
}
```

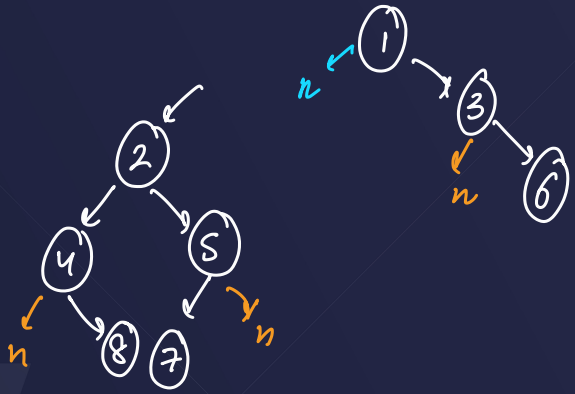



```
class Node {
    int val;
    Node left;
    Node right;
}
```

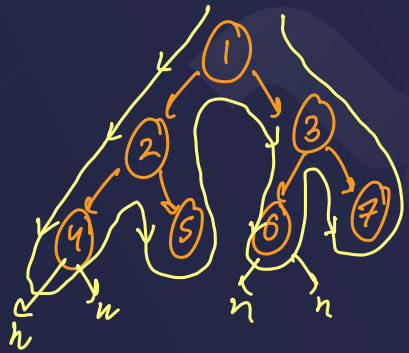

root
1
LST
2
3
RST
4
5
8
7
9


1
2
3
4
5
7


1
2
3
4
5
7


1
n
3
n
6
2
4
5
n
8
7
n

```java
public static void display(Node root){
    if(root==null) return;
    System.out.print(root.val+"-> ");
    if(root.left!=null) System.out.print(root.left.val+" ");
    if(root.right!=null) System.out.print(root.right.val+" ");
    System.out.println();
    display(root.left);
    display(root.right);
}
```

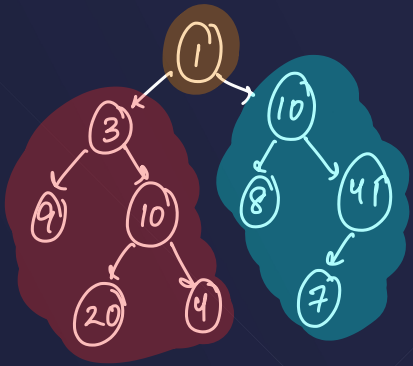


- 1 → 2 3
- 2 → 4 5
- 4 →
- 5 →
- 3 → 6 7
- 6 →
- 7 →

Sum = 45 ?

Sum(1) = 1 + Sum(2) + Sum(3)

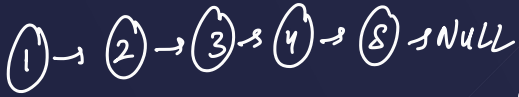
Sum(2) = 2 + Sum(4) + Sum(5)



max(root.val, sum(left), sum(right))

Integer.MIN_VALUE → java → -2^{31}

INT_MIN → C++



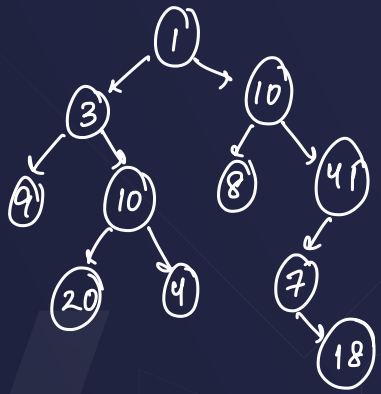
size & length one same

{1, 2, 3, 4, 5, 6}

but in trees size = number of nodes

height =

Levels / Height :



levels(1) = 1 + max(left levels, right levels)

H.W.

1) to calculate the 'min' value in tree

2) Product of all nodes