Trees 2

Madhan Kumar M S
Abhishek Sharma
Akansh Nirmal
Balaji S K
Bhaveshkumar
Burhan
Gagan Kumar S
Ishan
Khushi Raj
Murali Mudigonda
Naval Oli
Nikhil Pandey
Pankaj Bhanu
Purusharth A
Rajat Sharma
Rajendra
Sanket Giri
Saurabh Ruikar
Shani Jaiswal
sharath r
Shrikanth
Sneha Loganathan
Subhashini
Sumit Adwani
Suyash Gupta
Vasanth
Vetrivel H M
Yugesh v

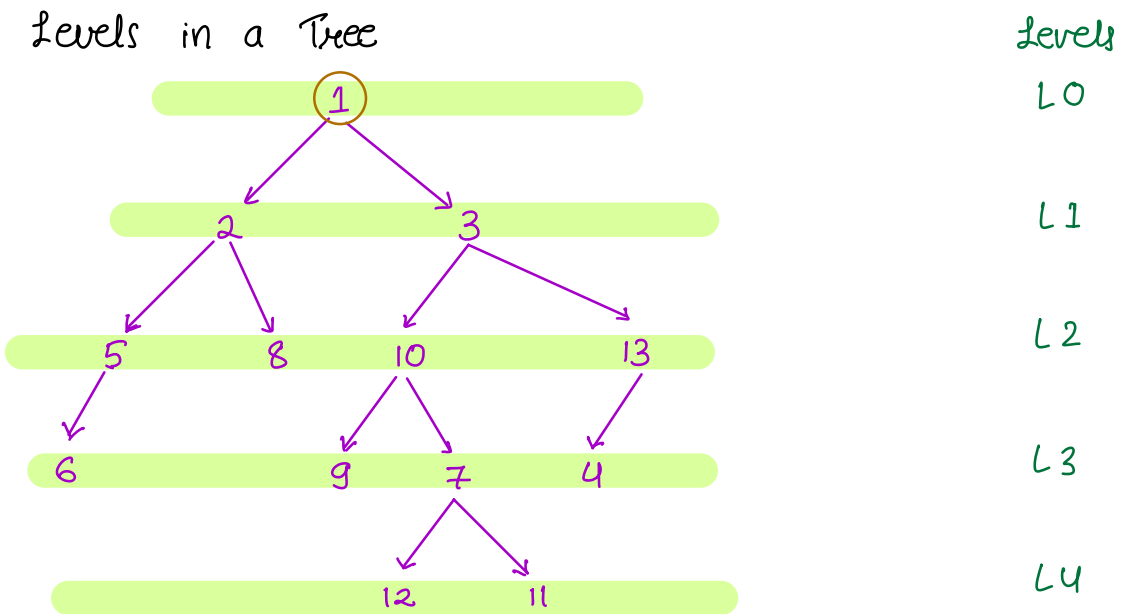
- Level Order Traversal
- Left view & Right view
- Vertical Order Traversal
- Top view & bottom view
- Types of Binary tree
- check if a binary tree is height balanced or not

Important Announcement

- Career Readiness 12 pm Sunday
- 11 am Sat → Doubt Solving

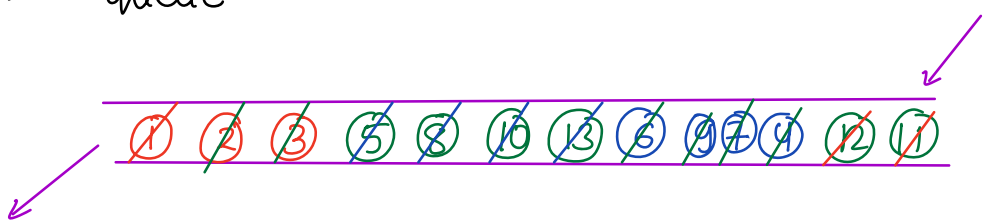
Two ptr, LL Stacks Queues

Level Order Traversal



Output : 1 2 3 5 8 10 13 6 9 7 4 12 11

FIFO ⟶ queue



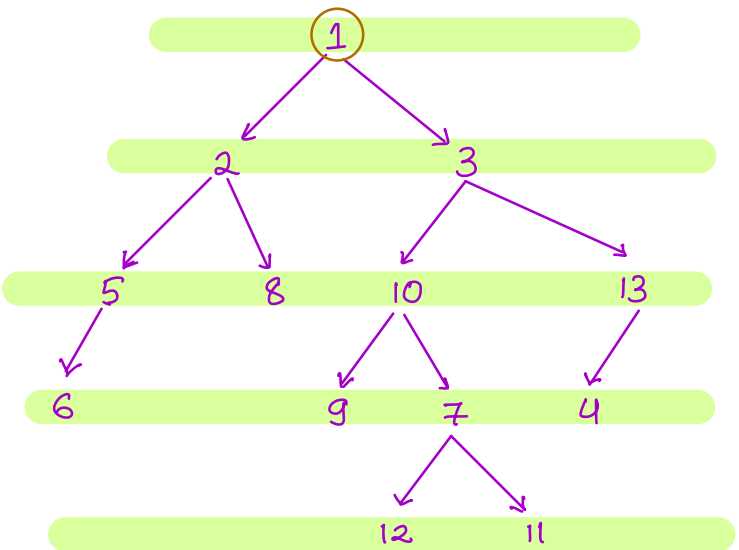
Output : 1 2 3 5 8 10 13 6 9 7 4 12 11

```
if( root == null ) return
q // initialise Queue of TreeNode
q.enqueue( root )

while ( ! q.isEmpty() ) {
```

```
    x = q.dequeue()
    print (x.data)
    if (x.left != null) q.enqueue (x.left)
    if (x.right != null) q.enqueue (x.right)
}
```

Print all the levels in a separate line.

Levels in a Tree



Levels

L0

L1

L2

L3

L4

Output :

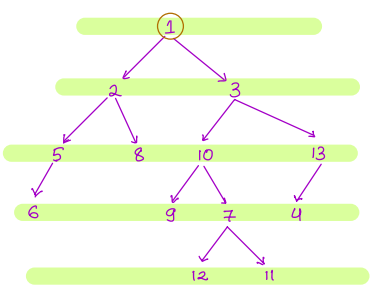
1

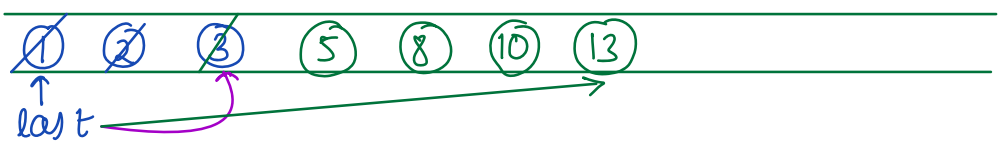
2 3

5 8 10 13

6 9 7 4

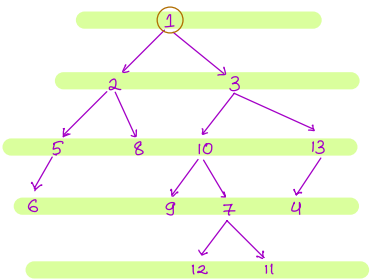
12 11





Output 1 '\n'
2 3 '\n'

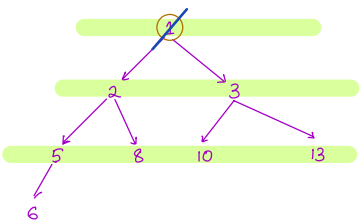
```
if( root == null ) return
q // initialise Queue of TreeNode
q.enqueue( root )
last = root
while ( ! q.isEmpty() ) {
    x = q.dequeue()
    print (x.data)
    if (x.left != null) q.enqueue (x.left)
    if (x.right != null) q.enqueue (x.right)

    if (x == last && !q.isEmpty()) {
        print('\n')
        last = q.rear()
    }
}
```



TC: O(N)

SC: O(N)

standard sol"

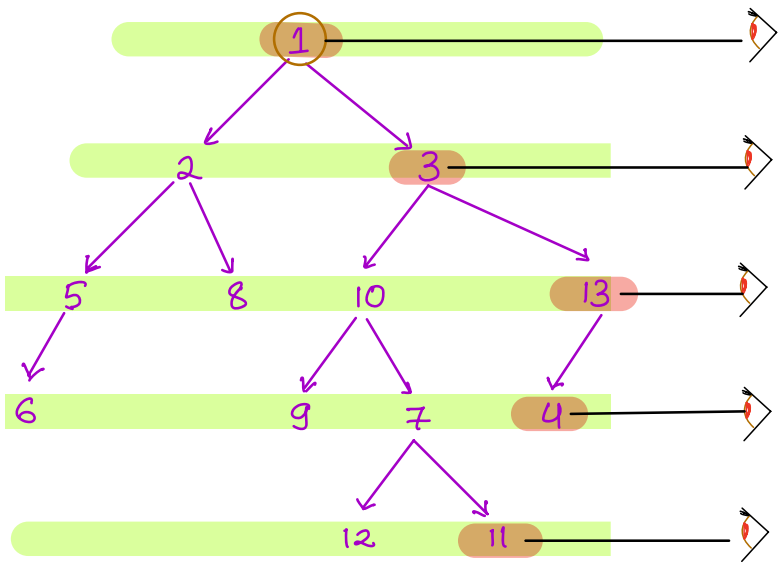


(1,0) (2,1) (3,1) (5,2) (8,2) (10,2) (13,2)

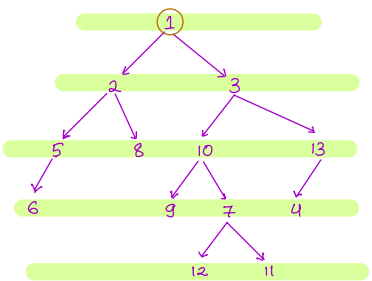
(6,3)

```
class Pair {
    int level;
    Treenode node;
}
```

Print right view of binary tree.

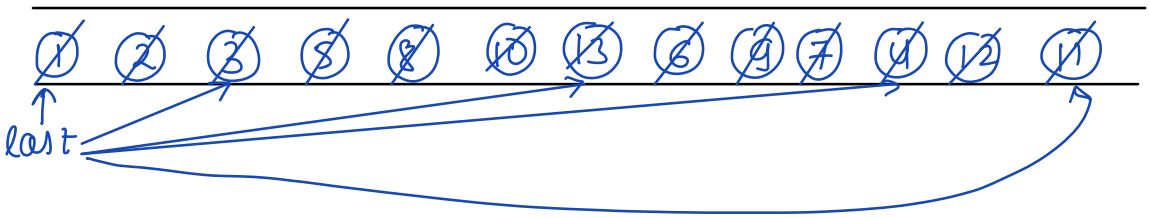


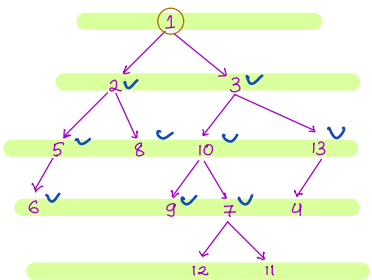

```
if( root == null ) return
q // initialise Queue of TreeNode
q.enqueue( root )
last = root
while ( ! q.isEmpty() ) {
        x = q.dequeue()
        if (x.left != null) q.enqueue (x.left)
        if (x.right != null) q.enqueue (x.right)

        if (x == last ) {
                print( x.data )
                if( !q.isEmpty() ) last = q.rear()
        }
}
```



TC: O(N)

SC: O(N)

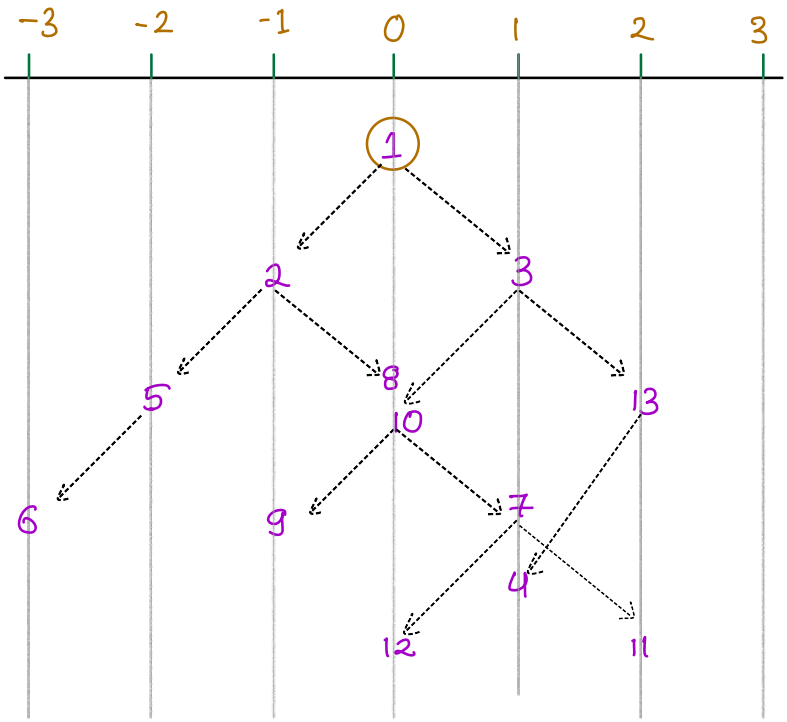




output 1 3 13 4 11

HW Print the left view of binary tree

Vertical Order Traversal



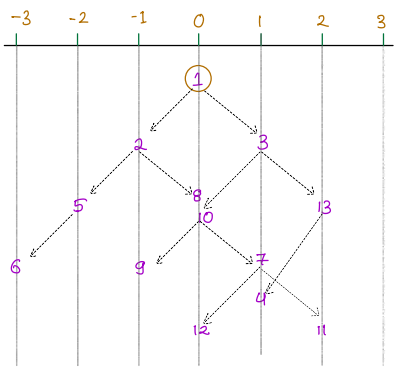
1> -1 +1

2> Print from top to bottom

3> Overlap : Give preference to left

output

6
5
2 9
1 8 10 12
3 7 4
13 11

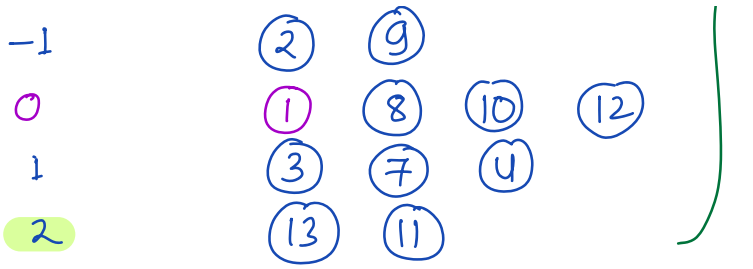


(1,0), (2,-1), (3,1) (5,-2), (8,0) (10,0) (13,2)

(6,-3) (9,-1) (7,1) (4,1) (12,0), (11,2)

V level	node
-3	6
-2	5

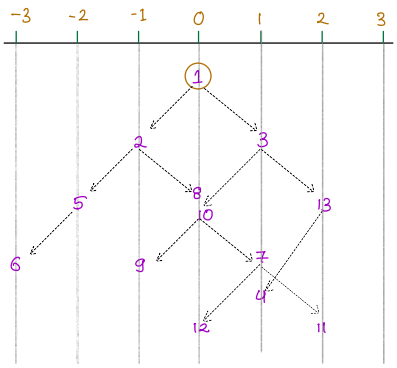
HM < Int, List <Node> >



Should we use TreeMap to maintain sorted vlevel?

NO → sortedMap

```
class Pair {
    TreeNode Node;
    int vlevel;
}
```




```
q // init Queue of Pair
q.enqueue(new Pair(root, 0))
hm // key ⟶ int   value ⟶ List of Integer
minL, maxL
while(!q.isEmpty) {
    p = q.dequeue()
    minL = min(minL, p.vlevel)
    maxL = max(maxL, p.vlevel)
    nodes = hm.getOrDefault(p.vlevel, new AL<>())
    nodes.add(p.node.data)
    hm.put(p.vlevel, nodes)

    if(p.node.left != null) {
        q.enqueue(new Pair(p.node.left, p.vlevel-1)
```

```
        }
        if (p.node.right != null) {
            q.enqueue(new Pair(p.node.right, p.vlevel+1))
        }
    }

    // go from minL to maxL
    for l ⟶ minL to maxL {
        print(hm.get(l))
    }
```

TC: O(N)

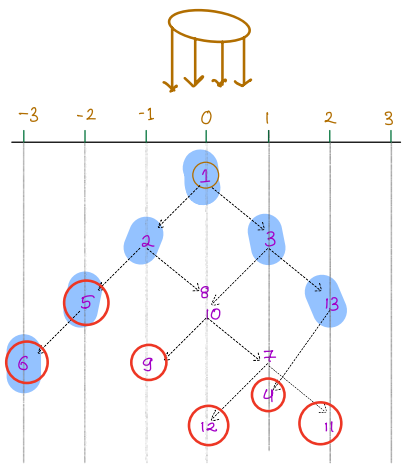
SC: O(N)

22: 54

Top view

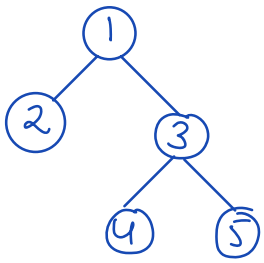
output

6 5 2 9 1 8 10 12 3 7 4 13 11



Bottom view ⟶ represents BV

Types of Binary Tree



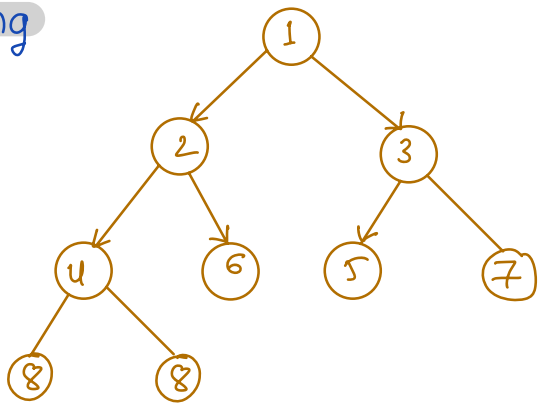
1> **Proper binary tree**

Every node has either 0 or 2 children

2> **Complete Binary Tree**

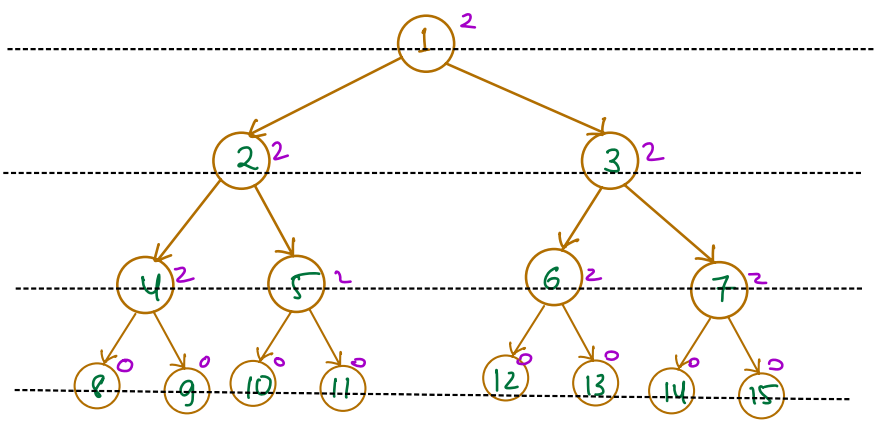
All levels filled, and last level maybe filled.

left to right filling

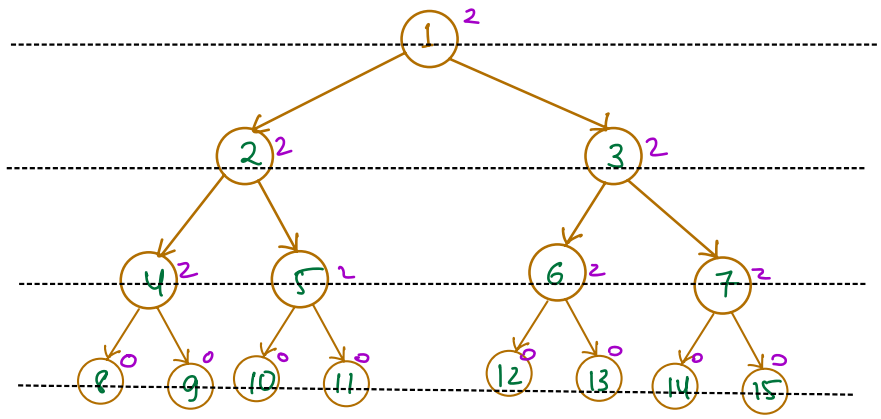


3> **Perfect binary tree** is **Proper & complete**

All the levels are completely filled.



What is the height of perfect binary tree with N nodes ?



Lets say height of tree = H

$$2^0 + 2^1 + 2^2 \ldots\ldots + 2^H = N$$

$$\frac{2^{H+1} - 1}{2 - 1} = N$$

$$2^{H+1} = N + 1$$

$$H + 1 = \log_2 N + 1$$

$$H = \log_2(N+1) - 1$$

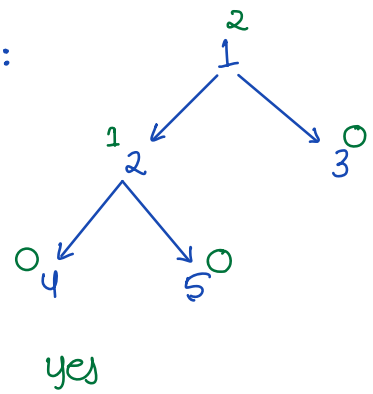
$O(\log(N))$

Check if the given tree is height balanced ?

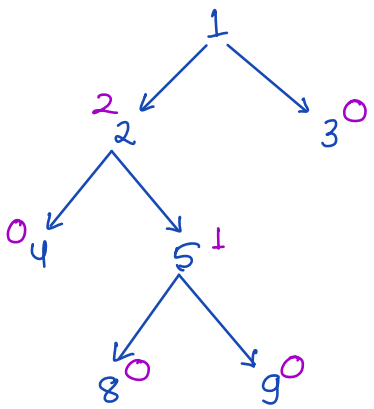
∀ nodes

$$\left| \text{height of left child} - \text{height of right child} \right| <= 1$$

Eg :

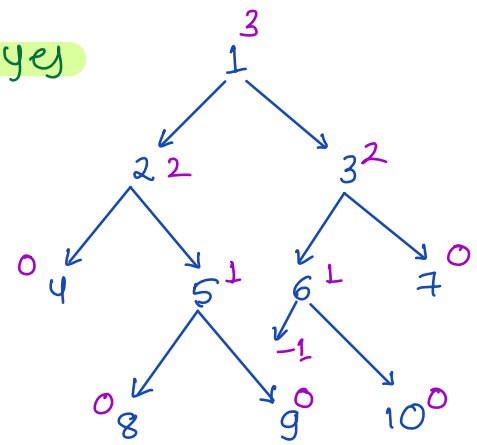


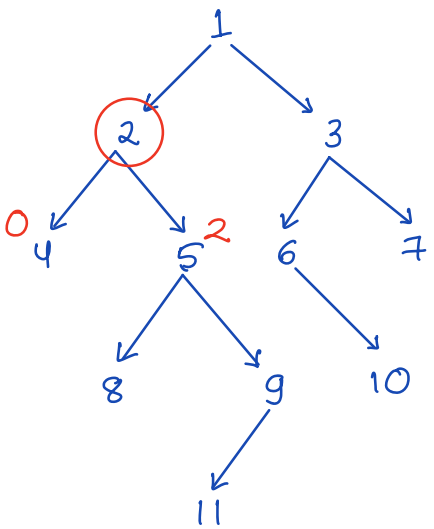
yes



NO

yes

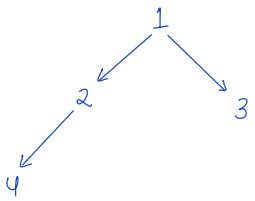




NO

Height (node) = $\left.\begin{matrix} \text{height (left child)} \\ \text{height (right child)} \end{matrix}\right\}$ max + 1

Traversal



balanced = true // global

```
int getH(root) {
    if(root == null) return -1
    lH = getH(root.left)
    rH = getH(root.right)

    if(abs(lH - rH) > 1) balanced = false

    return max(lH, rH) + 1
}

main() {
    balanced = true
    getH(root)
    print(balanced)
}
```

TC: O(N)

SC: O(H)