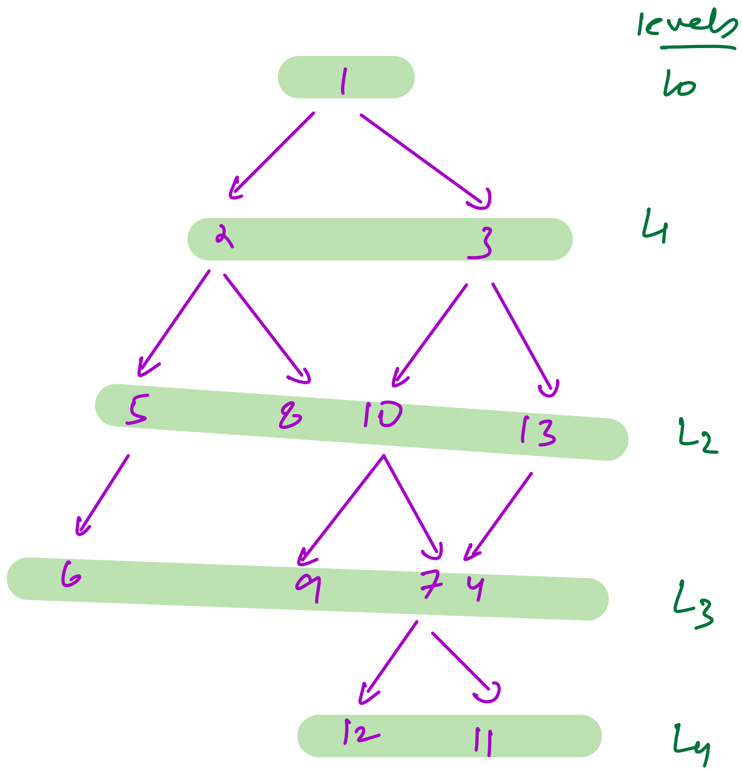
Trees 2 : Views & Types

Level-order traversal



O/P → 1

2 3

5 8 10 13

6 9 7 4

12 11

fifo → queue

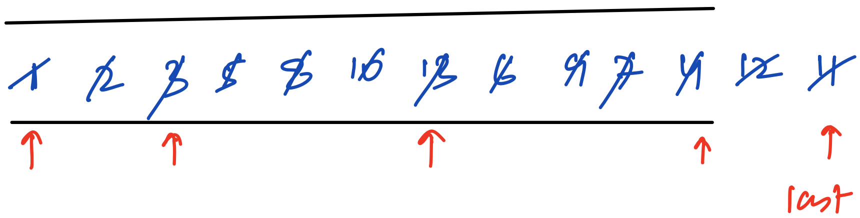
~~1~~ ~~2~~ ~~3~~ ~~5~~ ~~8~~ ~~10~~ ~~13~~ ~~6~~ ~~9~~ ~~7~~ ~~4~~ ~~12~~ ~~11~~

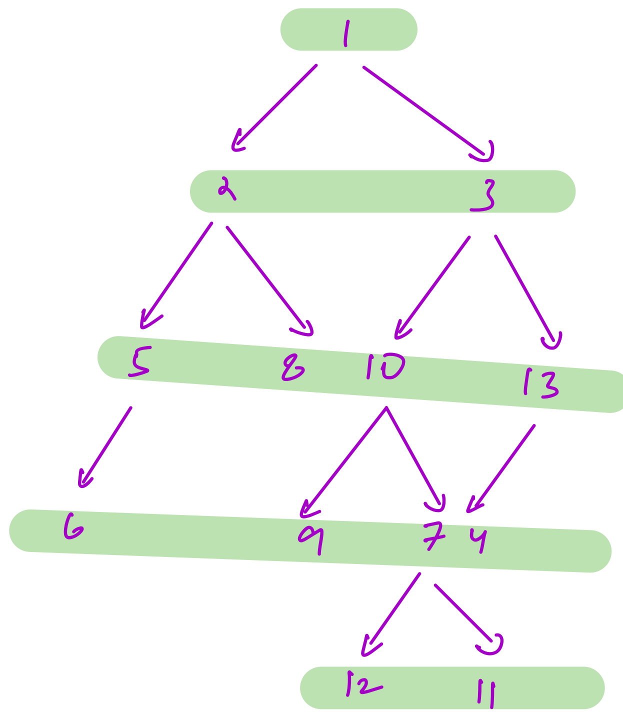
O/P : 1 2 3 5 8 10 13 6 9 7 4 12 11

code

```
if(root == NULL)   return
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)
}
```





o/p: 1

2 3

5 8 10 13

6 9 7 4

12 11

```
if(root == NULL)   return
q.enqueue(root)
last = root

while(!q.isEmpty()) {
    x = q.dequeue()
    print(x.data)

    if(x.left != NULL)   q.enqueue(x.left)
```

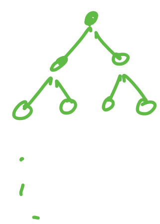
```
if(x.right != NULL)   q.enqueue(x.right)

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

}
```

TC = $O(N)$

SC = $O(N)$



2^0

2^1

2^2

$\vdots$

2^H max size of queue

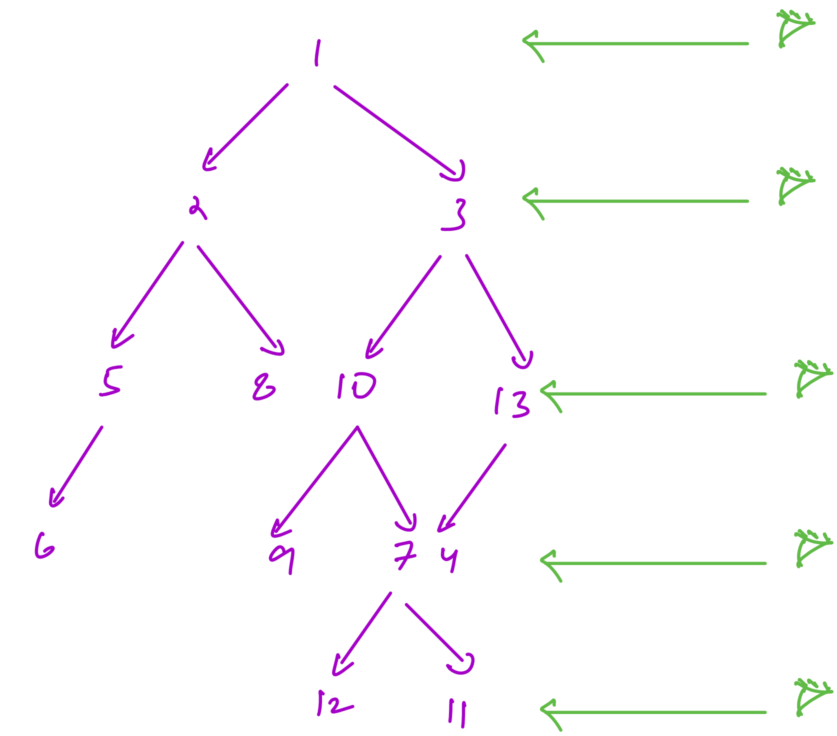
$$2^0 + 2^1 + 2^2 + \dots + 2^H = N$$

$$2^0\left(\frac{2^{H+1}-1}{2-1}\right) = N$$

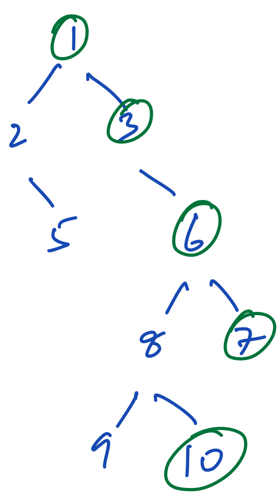
$$2^{H+1} = N+1$$

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

Question → Print right view of binary tree



O/P → 1 3 13 4 11



→ 1 3 6 7 10

Solution : Print last node of each level

Code

```
if(root == NULL)    return

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

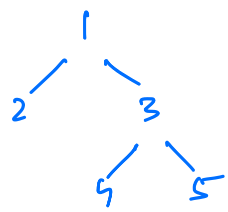
TC = O(N)

SC = O(N)

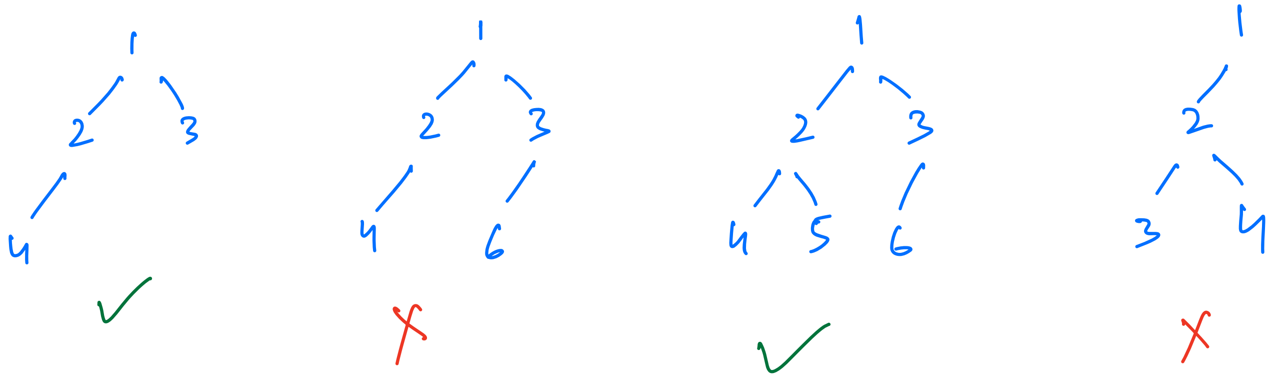
HW: print left view of BT

Types of binary tree (w.r.t structure)

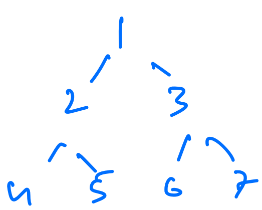
1. **Proper binary tree** → Every node has either 0 or 2 children

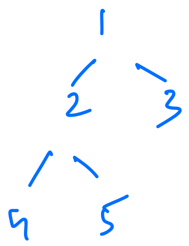


2. **Complete binary tree** → All levels are complete except maybe the last level which is filled from left to right.



3. **Perfect binary tree** → All levels are complete
(full binary tree)





Proper BT ✓

Complete BT ✓

Perfect BT ✗

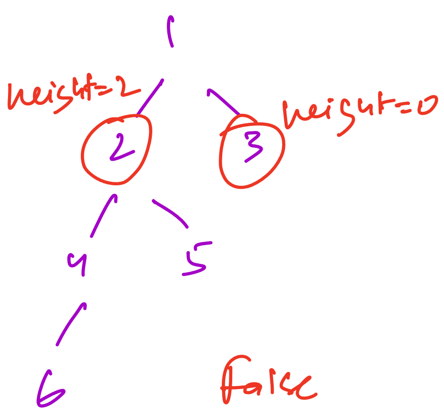
Question

Check if a given BT is height balanced?

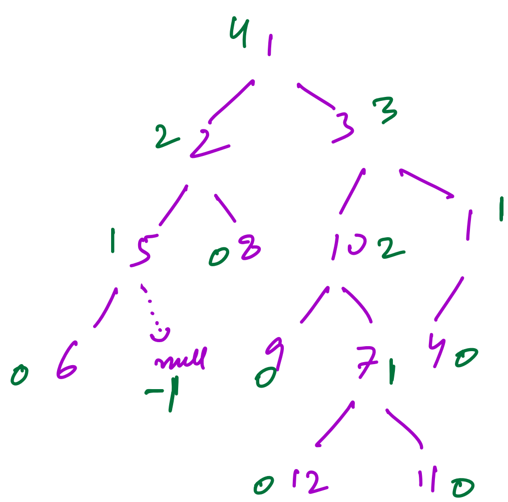
Height balanced tree :

$\forall$ nodes

$$\left| \text{Height of left subtree} - \text{Height of right subtree} \right| <= 1$$



False



True

Height(node) = max(Height(left), Height(right)) + 1

```
int height( root) {
    if ( root = = null)   return -1
    return max(height( root.left), height(root.right)) +1
}
```

```
bool isHeightBalanced( root) {
    if( root == null)   return true

    L = height( root.left)
    R = height( root.right)

    if( abs(L-R) >1 )
        return false

    return isHeightBalanced( root.left) &&
               isHeightBalanced( root.right)
}
```

→ for each node we are calling height() function

$TC = O(N^2)$

$SC = O(H)$

Optimize : modify the height function

isBalanced = true

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

    L = height(root.left)
    R = height (root. right)

    if( abs( L-R)  > 1)   isBalanced = false

    return  max( L, R) + 1

}
```

TC = O(N)

SC = O(H)

Question

Construct a BT from its inorder & postorder traversals.

All elements are unique

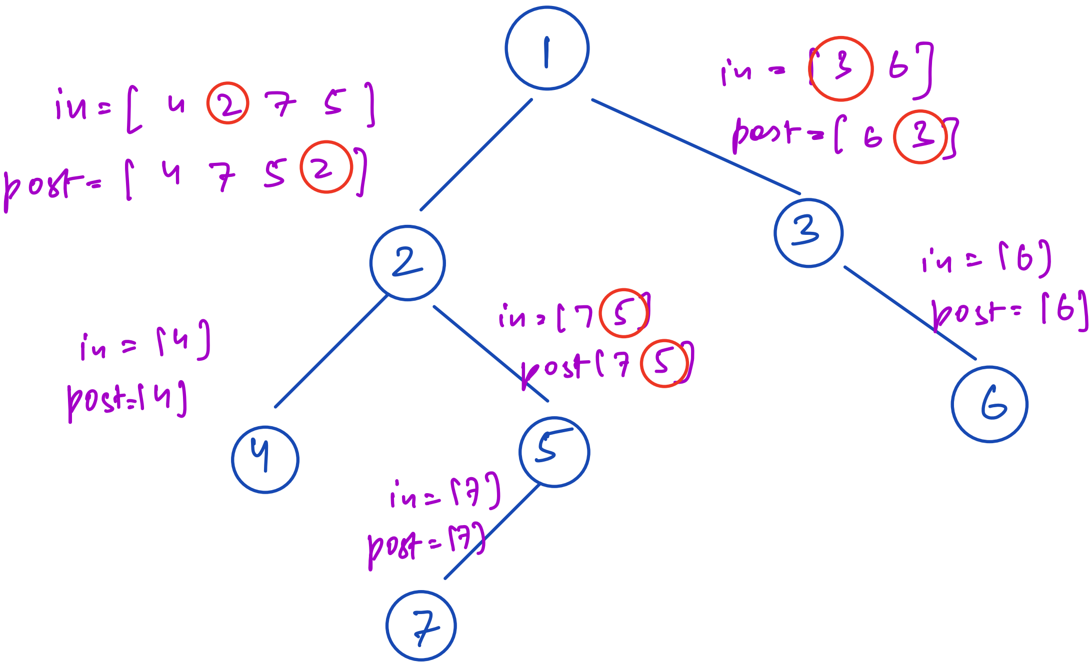
left subtree   right subtree

inorder = [4 2 7 5 1 3 6]

postorder = [4 7 5 2 6 3 1]

0 1 2 3 4 5 6

root



Code

```
Hashmap<int, int> hm
Node buildTree( in , post ) {

    n = in.size()

    for ( i = 0 to n-1 ) {
        hm [ in[i] ] = i
    }
    return build( in, 0, n-1, post, n-1 )
}

Node build ( in, i, j, post, k ) {

    if ( i > j )   return NULL

    root = new Node ( post [k] )
```

in_index = hm[post[K]]

size_rightSubtree = j - in_index ;

root.left = build(in, i, in_index - 1, post, K - 1 - size_rightSubtree)

root.right = build(in, in_index + 1, j, post, K-1)

return root

}

TC = O(N)

SC = O(N)

hashmap + recursion