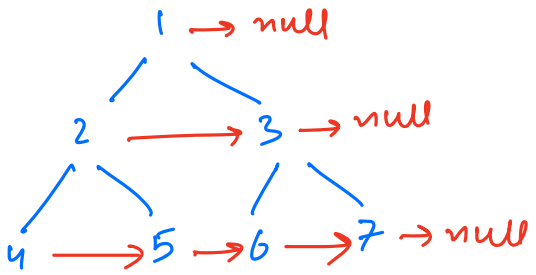
Trees 5 : Problems on trees

Question 1

(all levels completely filled)

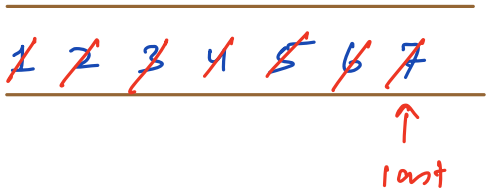
Given a perfect binary tree with next pointer in all nodes initially pointing to null.

Update the next pointer to point to next node in same level of nodes.



level order traversal

queue



Code

```
last = root
q.enqueue(root)
while ( !q.isEmpty() ) {
    x = q.dequeue()
    if ( x.left != null )   q.enqueue(x.left)
```

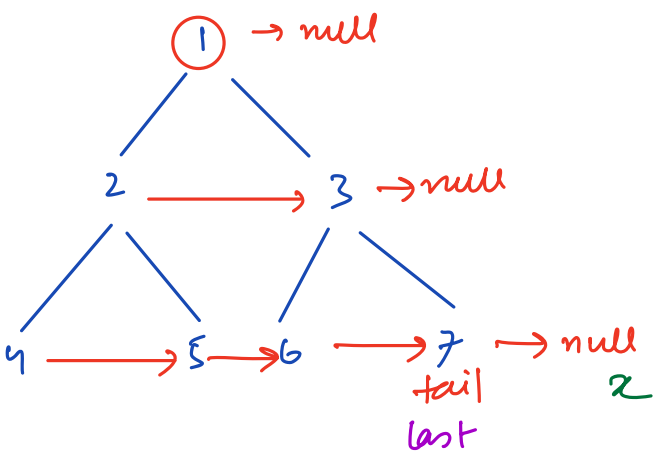
```
if(x.right != null)   q.enqueue(x.right)

if( x != last) {

      x.next = q.front()

}
else {

    if(!q.isEmpty())   last = q.rear()

}

}
```

TC = O(N)

SC = O(N)

~~queue~~ → OU ?

queue → linkedlist



Code

```
tail = root
last = root
x = root
while ( x != null ) {
    if ( x.left != null ) {
        tail.next = x.left
        tail = tail.next
    }
    if ( x.right != null ) {
        tail.next = x.right
        tail = tail.next
    }
    if( x != last )    x = x.next
    else {
        x = x.next
        last.next = null
        last = tail
    }
}
```

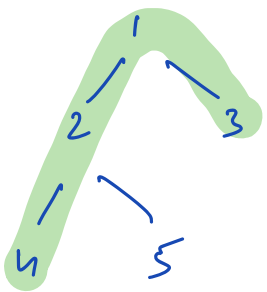
TC = O(N)

SC = O(1)

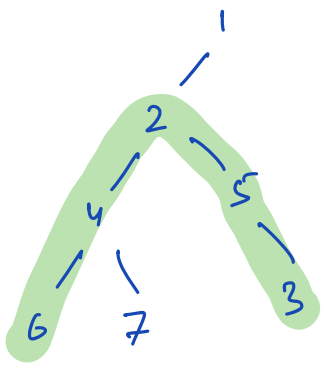
Question 2

Given a binary tree, find the length of the longest path b/w any 2 nodes in the tree.
The path may or may not pass through the root.

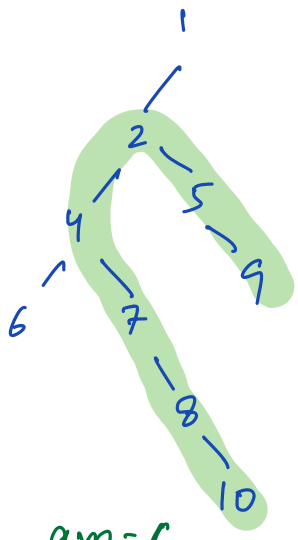
Diameter of binary tree : # edges in the longest path b/w any 2 leaf nodes.



ans = 3



ans = 4



ans = 6

Code

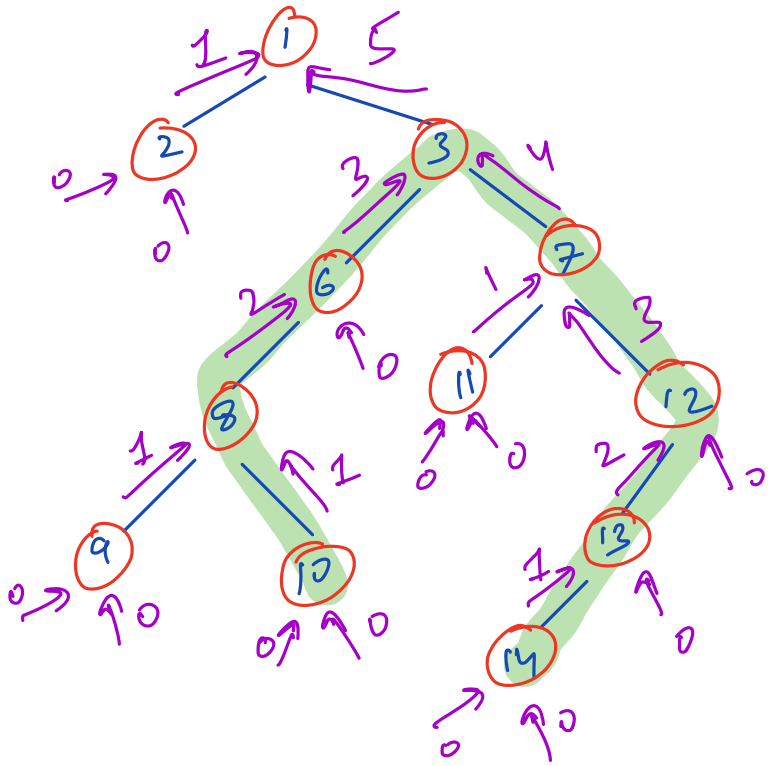
```
ans = 0
int height (root) {
        root == null
    if (!root)   return 0

    L = height (root.left)
```

```
R = height (root.right)

am = max (am, L+R).

return max(L,R) + 1

}
```

TC = O(N)

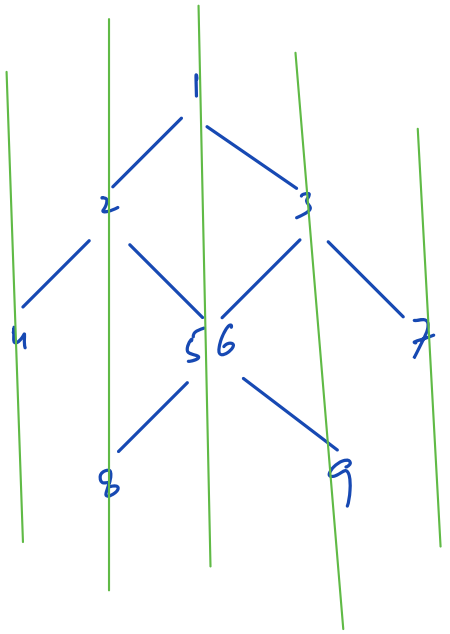
SC = O(H)

am = 0 ~~2~~ ~~4~~ 7



Question 3

Print vertical order traversal



o/p →

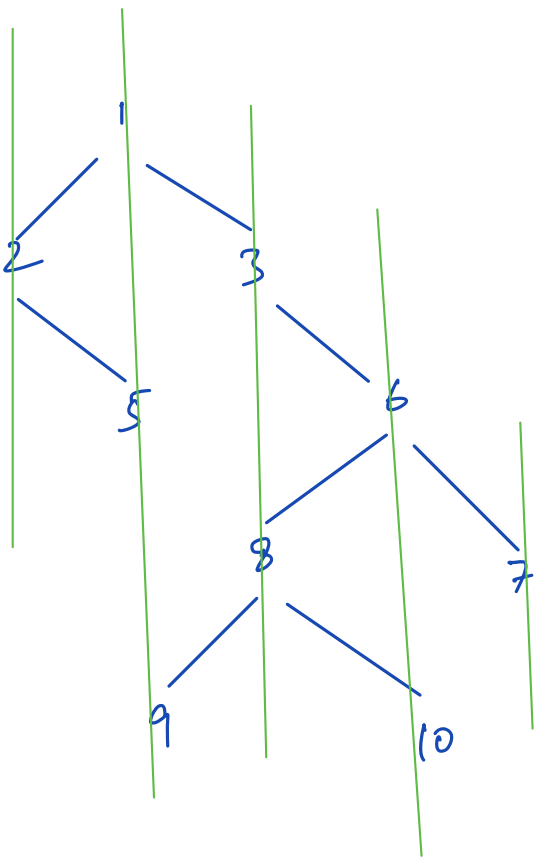
4

2 8

1 5 6

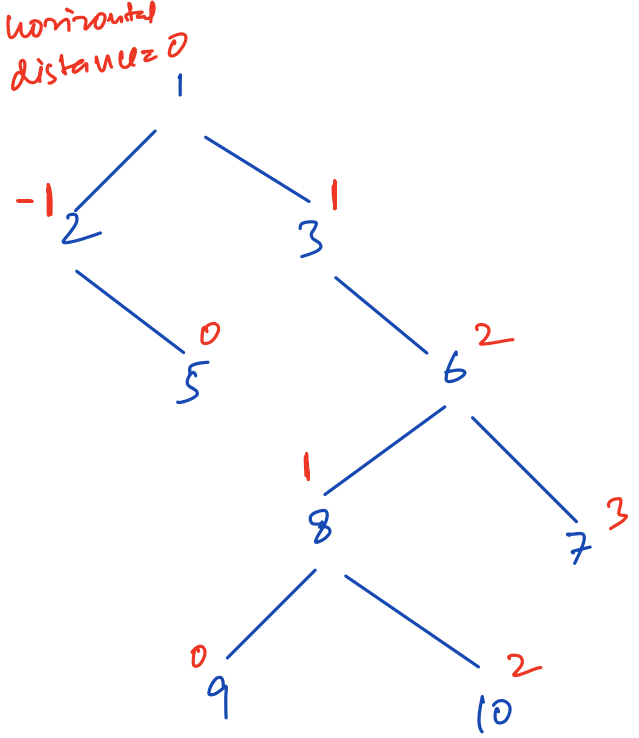
3 9

7



o/p → 2 1 5 9 3 8

6 10 7



Create hashmap with

key → horizontal distance

value → list of nodes

code

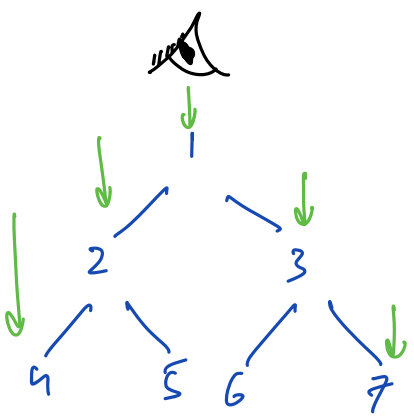
```
Hashmap < int, list<int> > hm

Queue < pair < Node, int > > q;

q.enqueue ( < root, 0 > );

mind = 0 , maxd = 0

while ( ! q.isEmpty() ) {

    x, d = q.dequeue()

    mind = min( mind, d )

    maxd = max( maxd, d )

    hm[d].add( x.data )
```

```
        if (x.left != null) {
            q.enqueue(<x.left, d-1>)
        }
        if (x.right != null) {
            q.enqueue(<x.right, d+1>)
        }
    }
    for (i = mind to maxd) {
        for (val in hm[i])
            print(val)
    }
```

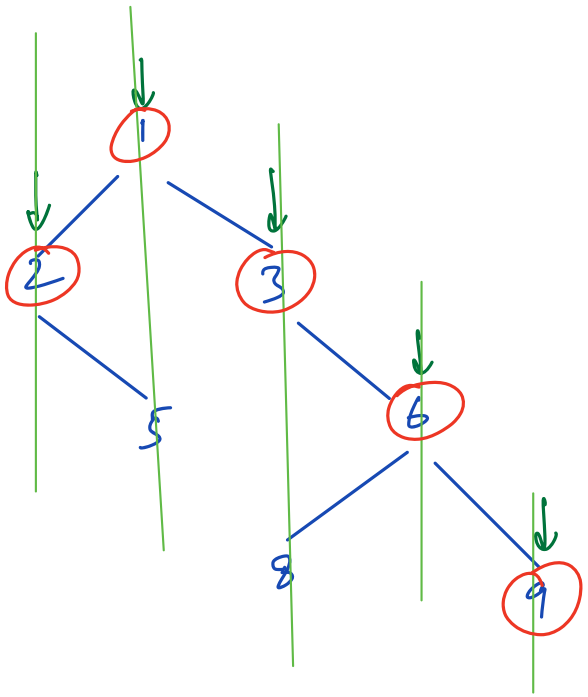
TC = O(N)

SC = O(N)

Question 4 Print top view



o/p : 4 2 1 3 7



o/p : 2 1 3 6 9

Observation : first element of each vertical in the vertical order traversal is answer.

Code

```
Hashmap < int, int >  hm

Queue < pair < Node, int > > q;

q.enqueue ( < root, 0 > );

mind = 0 , maxd = 0

while ( ! q.isEmpty() ) {

    x, d = q.dequeue()

    mind = min( mind, d )

    maxd = max( maxd, d )
```

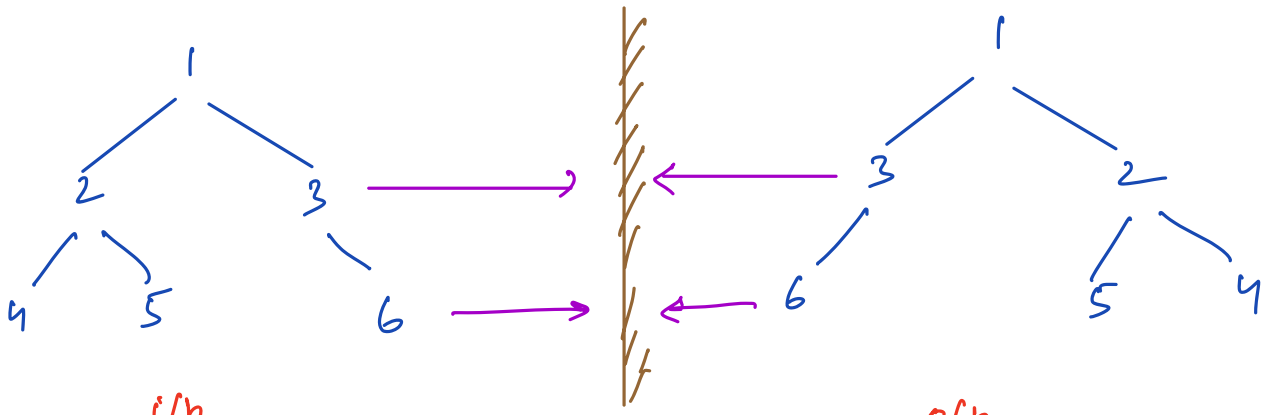
```
if (!hm.contains(d)) {
    hm[d] = x.data
}
if (x.left != null) {
    q.enqueue(<x.left, d-1>)
}
if (x.right != null) {
    q.enqueue(<x.right, d+1>)
}
}
for (i = mind to maxd) {
    print(hm[i])
}
```

TC = O(N)

SC = O(N)

Question 5

Invert the given binary tree



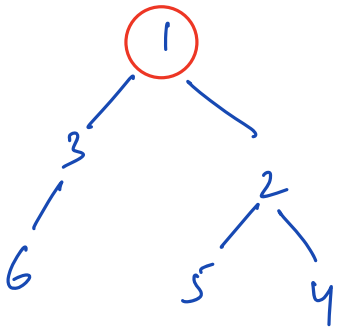
Solution : if node, swap left & right child

Code

```
Node invert(root) {
    if(!root) return null
    inverted_left = invert(root.left)
    inverted_right = invert(root.right)
    root.left = inverted_right
    root.right = inverted_left
    return root
}
```

TC = O(N)

SC : O(N)



✗
```
root.left = invert(root.right)
root.right = invert(root.left)
```

✓
```
temp = invert(root.right)
root.right = invert(root.left)
root.left = temp
```