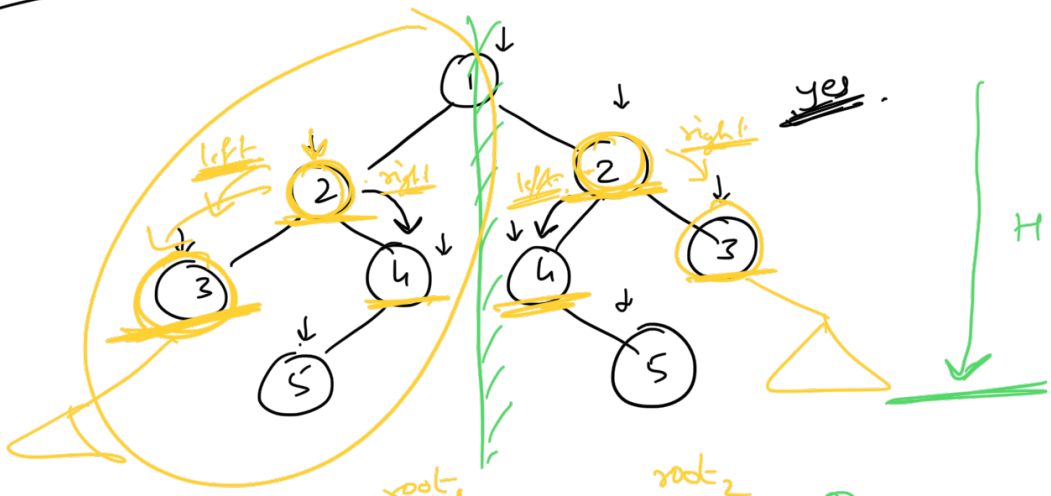
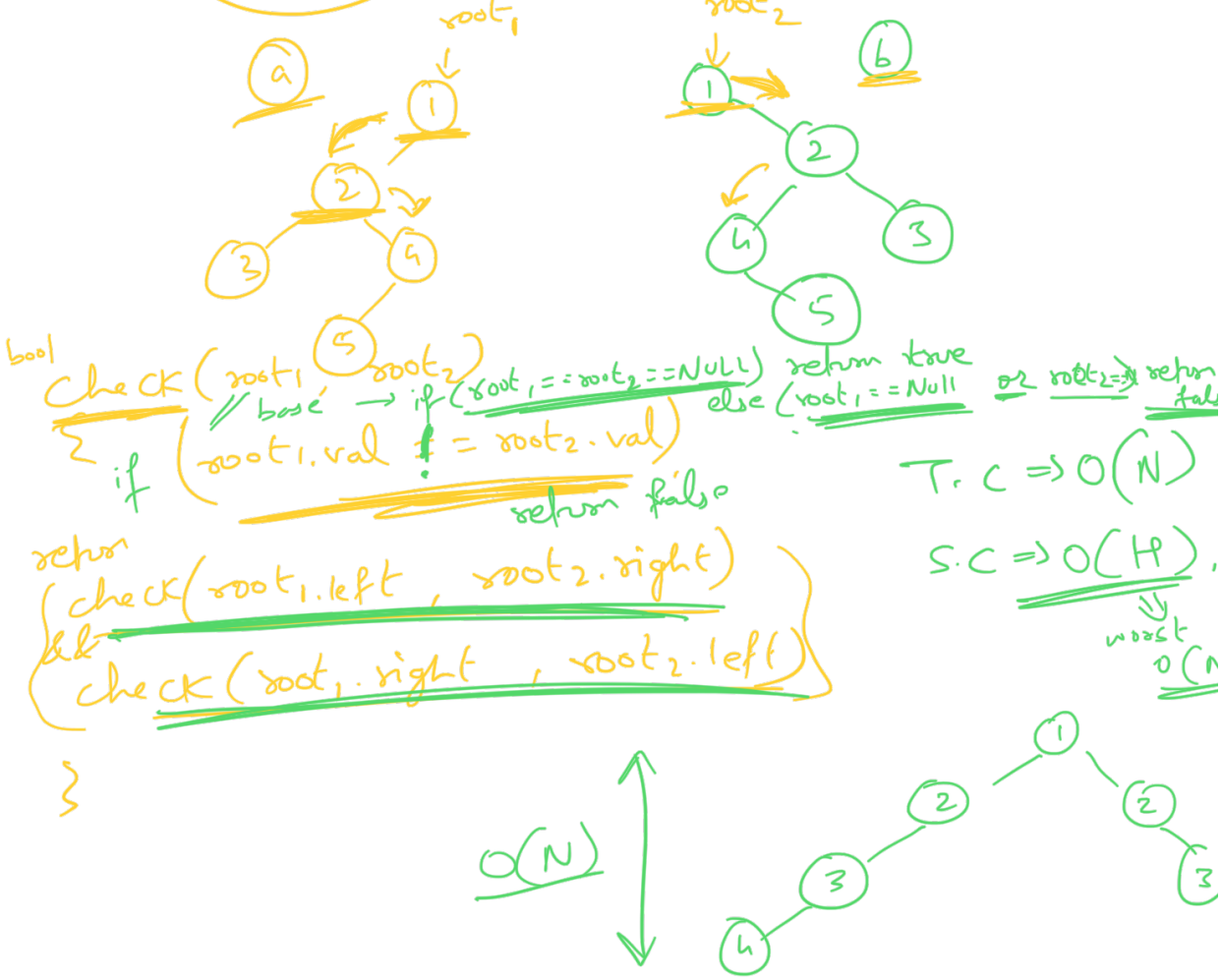
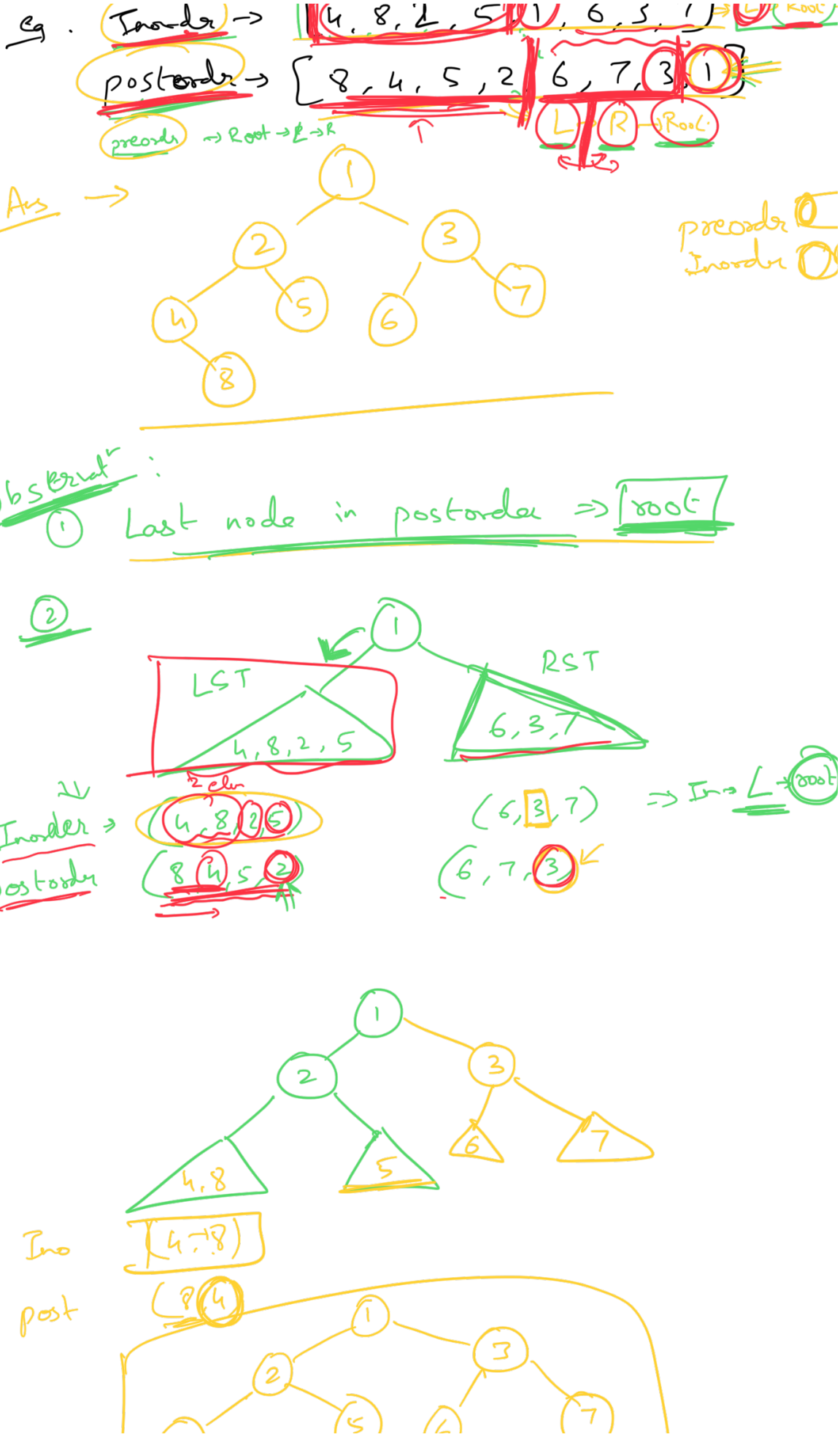
Q.1 Given a B.T., check if its symmetric.

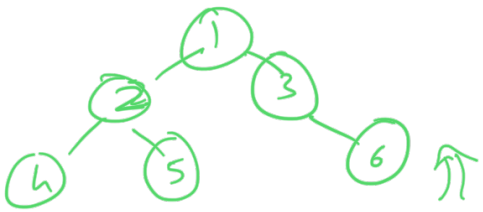


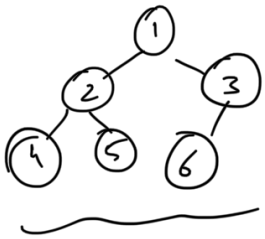


Q.2. Given postorder & inorder traversal, construct the B.T. (distinct)


Eg.
Inorder →
postorder →
preorder → Root → L → R
L
R
Root
Ans →
preorder
Inorder
Observation :
1
Last node in postorder ⇒ root
2
LST
RST
4,8,2,5
6,3,7
2 elm
Inorder →
postorder
(4,8,2,5)
(8,4,5,2)
(6,3,7)
(6,7,3)
⇒ In → L → root
4,8
6
7
5
Ino
(4,8)
post
(8,4)

Q.3 Given postorder & preorder, construct B.T.





↔ pre → 1, 2, 4, 5, 3, 6

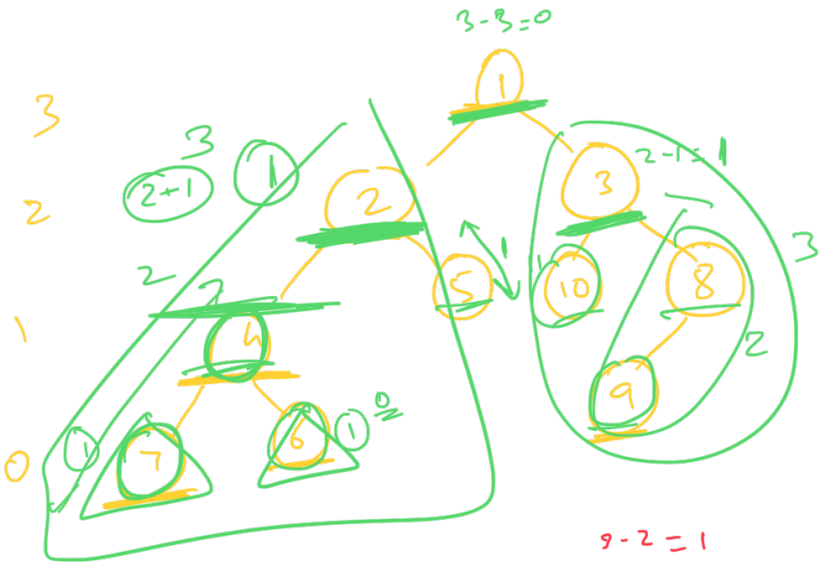
→ post → 4, 5, 2, 6, 3, 1

There is no unique B.T. you can constru

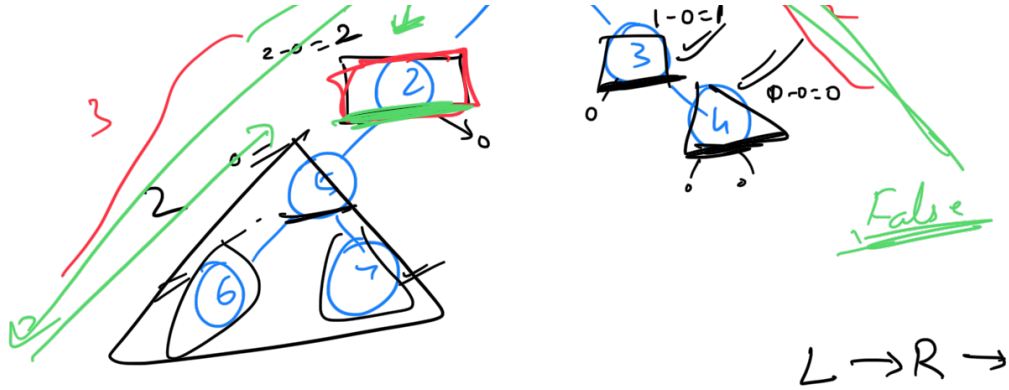
Q.4 Given B.T., check if tree is 'height' balanced.

Amazon Google FB Yahoo Adobe MS

For all nodes, (diff (height of L.S.T & height of R.S.T)) ≤ 1



|(LST - RST)|



L → R → root (postorder!)

Brute: For every node,
get height of LST
get height of RST
=> check $|(h_{LST} - h_{RST})| \leq 1$

bool isBalance(root) T.C

```
{ //Base case
  if(root==NULL) return True.

  H_LST = getHeight(root.left)   => O(H)
  H_RST = getHeight(root.right)  => O(H)
  return (|H_LST - H_RST| <= 1) && (isBalance(root.left)
                                 && isBalance(root.right))
}
```

T.C => O(N*H)

$LST' = \max(LST, RST) + 1$

---

struct Info { int height; bool isbal; }

class info { }

← check(root) isbal.

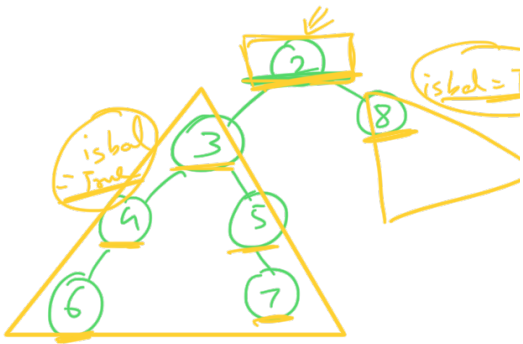
pair<int, bool>

```
Info check(root)
{
  //Base if(root==NULL) return info(0, true)

  Info LST = check(root.left)
  Info RST = check(root.right)
  if(left.isbal && right.isbal &&
     |left.height - right.height| ≤ 1)
    return Info(max(left.height, right.height)+1, true)
  else return Info(0, false)
}
```

T.C ⇒ O(N)

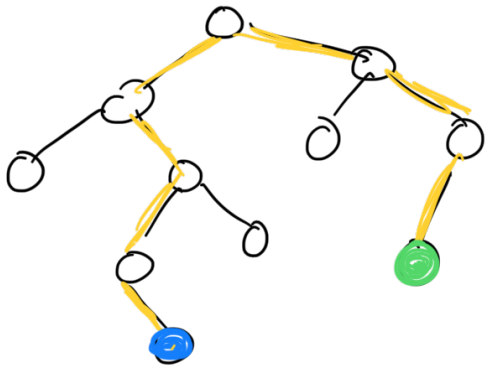
S.C ⇒ O(H).



Q.5. Find diameter of the tree

⇓

longest path b/w any two nodes.

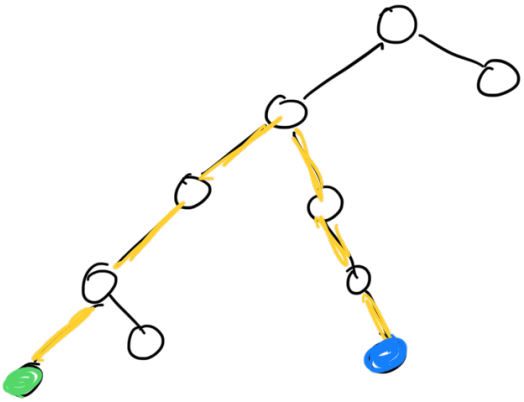


length = 7

Obs: (left height + right height)

max(LST) + max(RST)

wrong



① 2 nodes are alwg going to be leaf no

② Cases possible getDiameter(root)

max

a → (max(LST) + max(RST),

getDiamet(root.left)

(b) → get Diameter (root.right)

(c) →

int getDiameter (root)

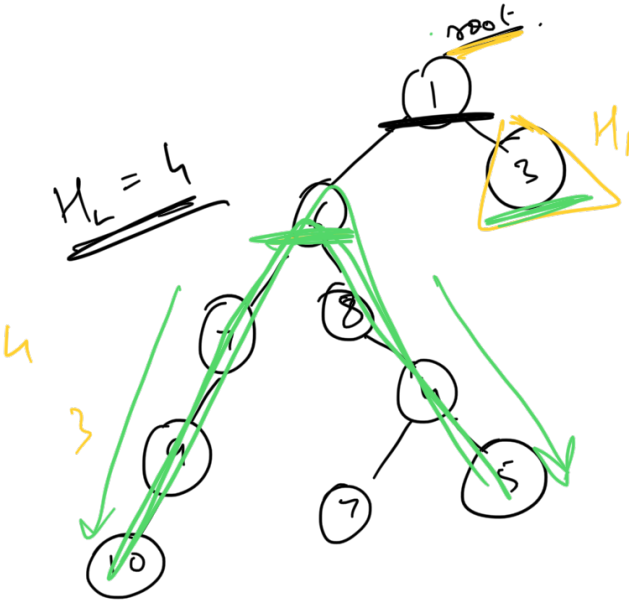
{ // Base if (root == null) return 0;

H_L = getHeight (root.left)

H_R = getHeight (root.right)

return max ($H_L + H_R$, getDia (r.left) , getDiam (r.right))

T.C => O(N*H)



getDia(1)

{ max ((4+1) , getDia(6) => 6 , getDia(3) 0)

max (5 , 6 , 0)

}

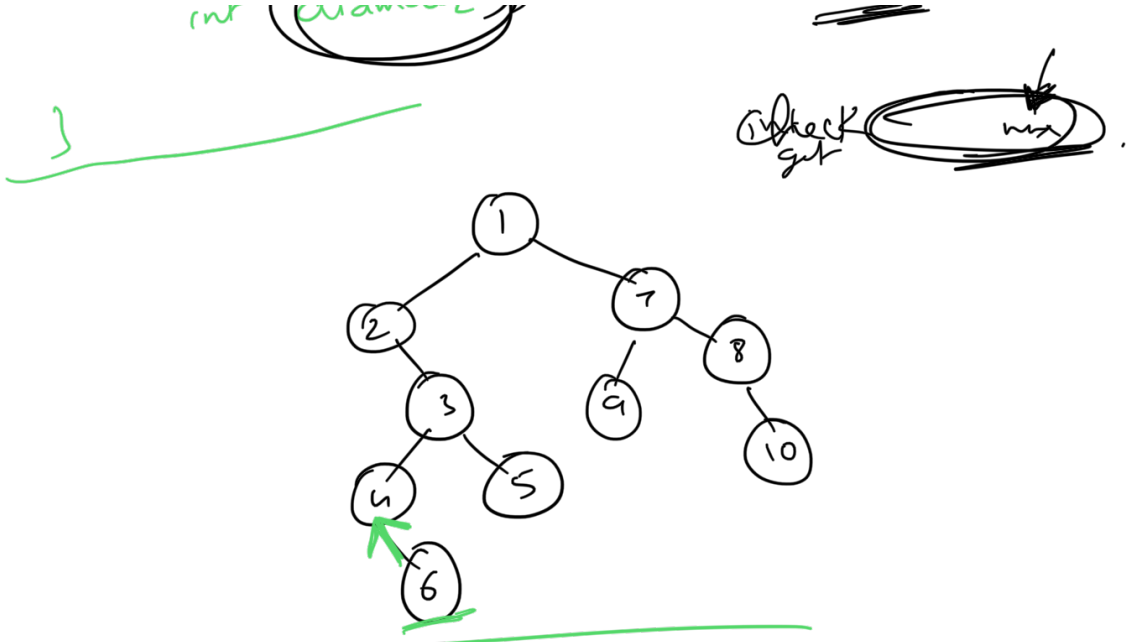
{ 2+3 , getDia(7) 2 , getDiam(8) 3 } + [6, 2, 3]

struct Info

{ int height, diameter } O(1)

int

2 int



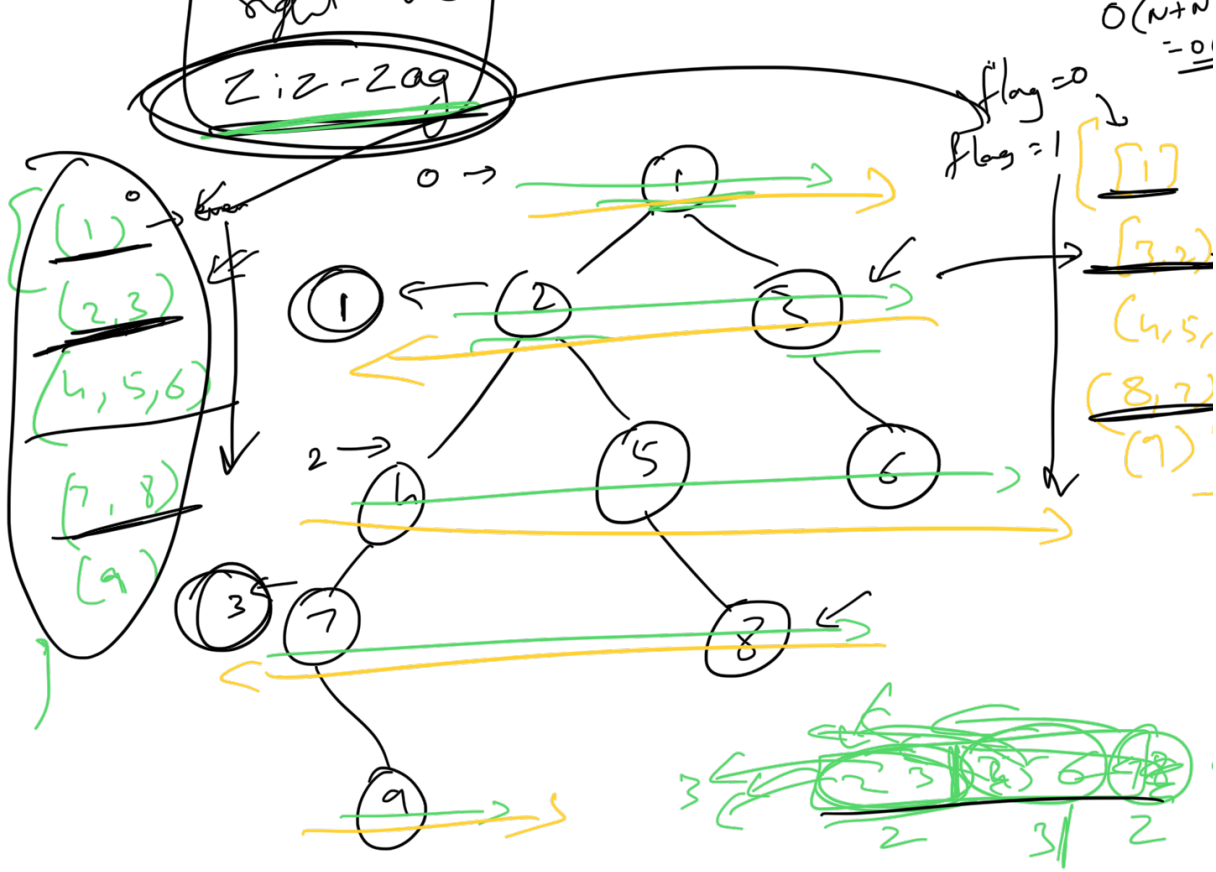
Star 10:25pm

Q.6 Level order traversal

Left view

right view

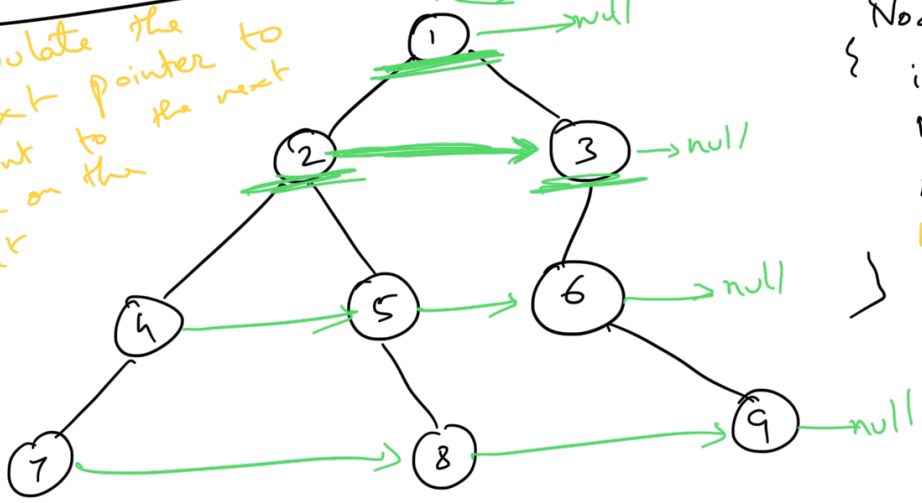
Zig-Zag



vertical

Doubts / extra problems

Populate the next pointer to point to the next node on the right

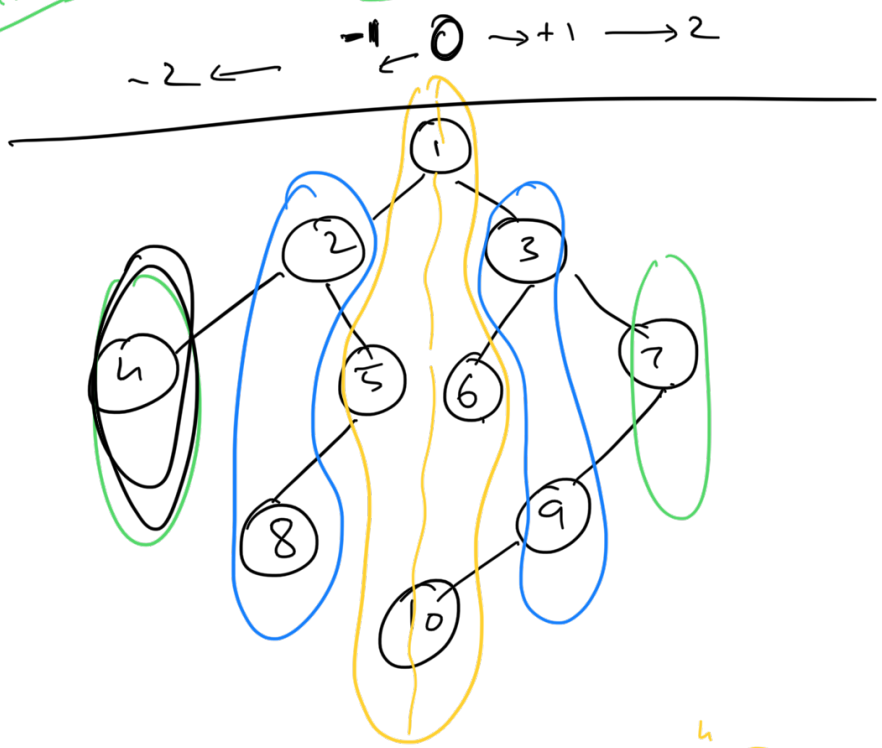



```
Node
{ int data,
  Node left,
  Node right,
  Node next;
}
```

O(N)

Method 1 queue

Method 2 → O(1) space



(4)
(2,8)
(1,5,6,10)
(3,9)
(7)