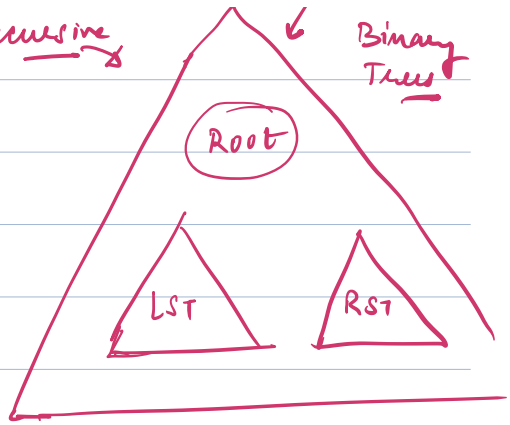
Recap :

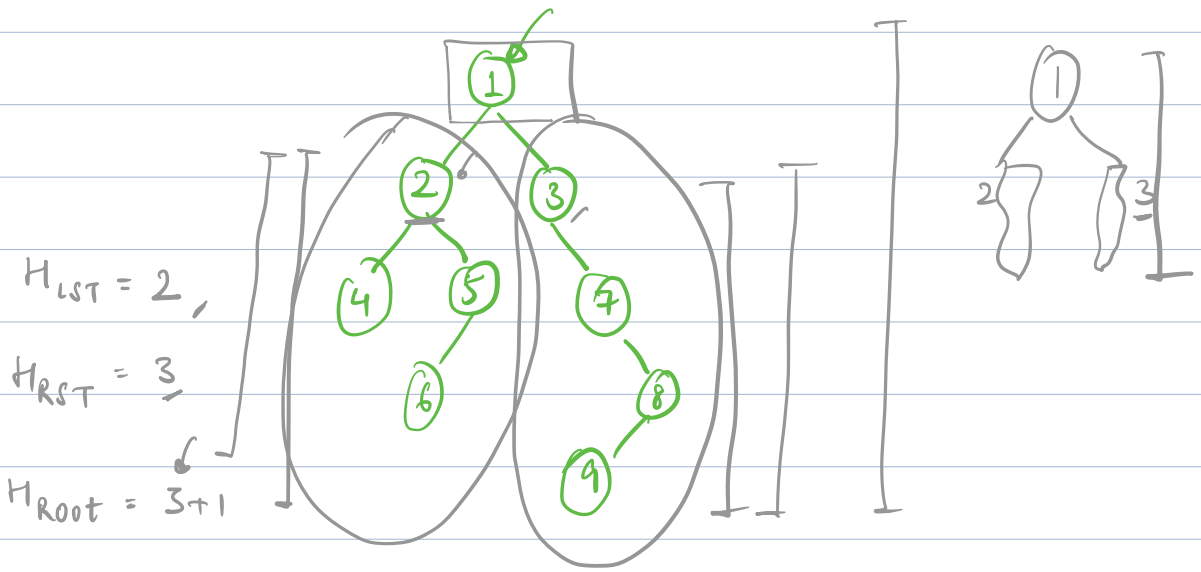
1. Tree → terms
2. Structure of Tree :
3. Traversals
   - → Level order (Queues)
   - → Post, Pre, In order (Recursive)

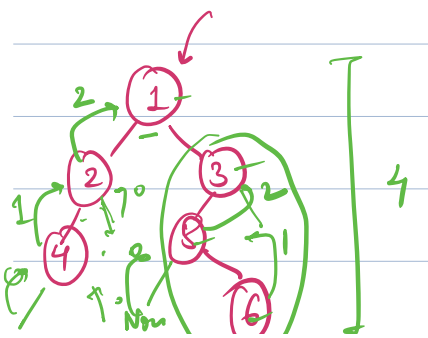


→ How to find height of the tree?

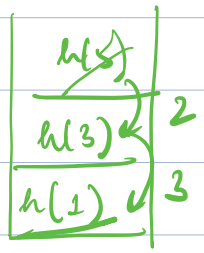
⇓

Distance from the farthest leaf node.



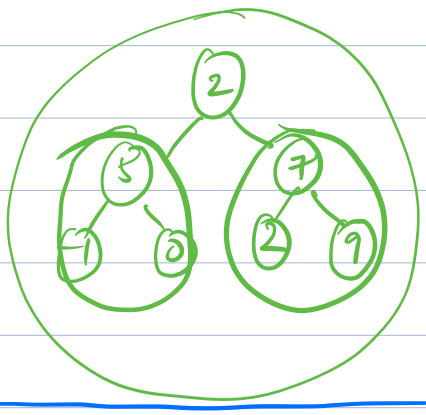



```
def height ( root ):
    # Assumption : This fn returns
      height of tree rooted at root.
    if root == None:
        return 0
```



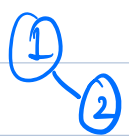
h_lst = height (root.left)
h_rst = height (root.right)
return max (h_lst, h_rst) + 1

Q Sum of all nodes of a tree → given the root node




```
def sum (root)
    if root == None:
        return 0
    return root.data + sum(root.left)
                     + sum(root.right)
```

Q3 Given root nodes of 2 binary trees
Return true if both the trees are identical



X

X

def is_identical (root1 , root2)

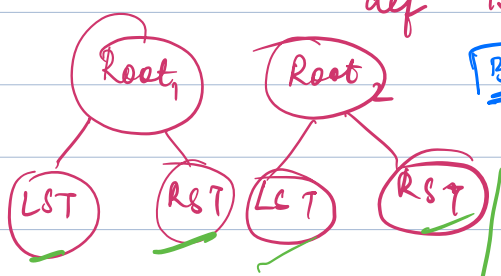
BC if root1 == None and root2 == None
return True

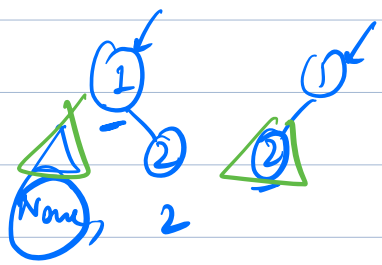
→ if root1 == None or root2 == None
return False

is_left_matching = is_identical (
root1.left, root2.left)

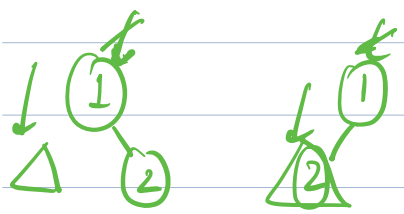
is_right_matching = is_identical (
root1.right, root2.right)

return is_left_matching and
is_right_matching
and root1.data == root2.
data



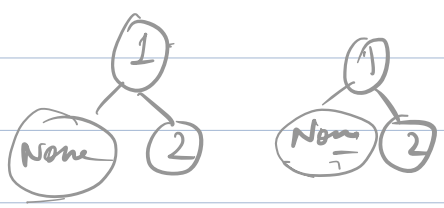


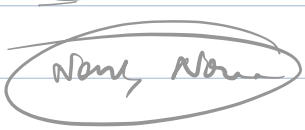
ML



i

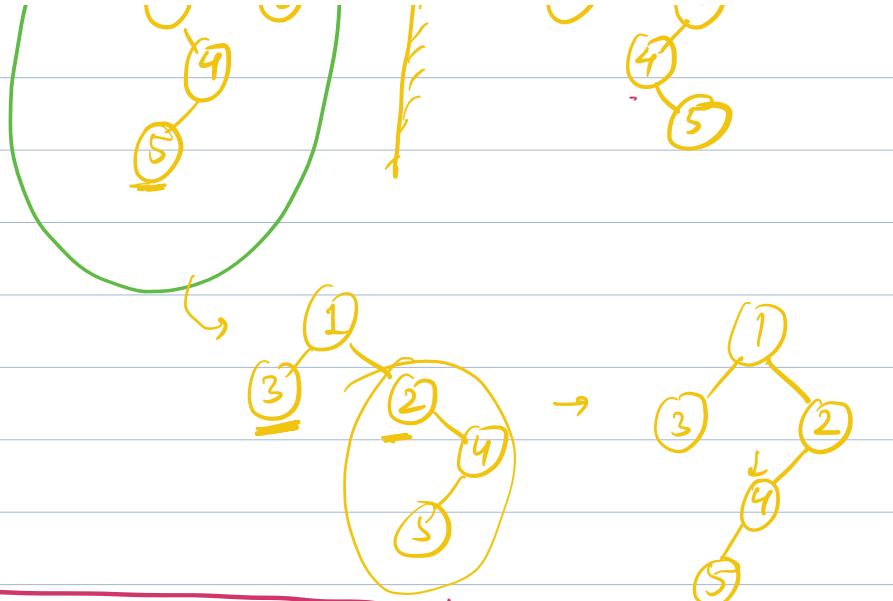
False



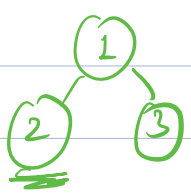


Q Given a binary tree.
Invert the tree and return the root Node

2 mins



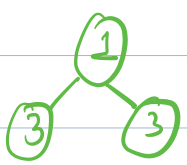
```
def invert ( root ) :
    if root == None :
        return root.
    tmp = root.left.
    root.left = invert(root.right)
    root.right = invert ( tmp )
    return root.
```

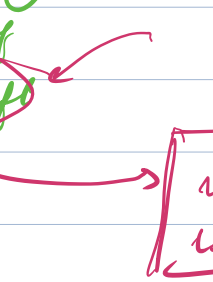


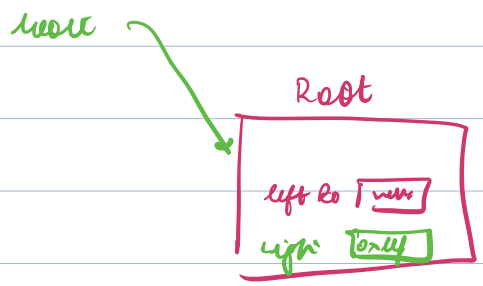
invert(1)

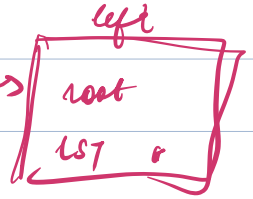
1.left = 3.

1.right = invert (1.left)



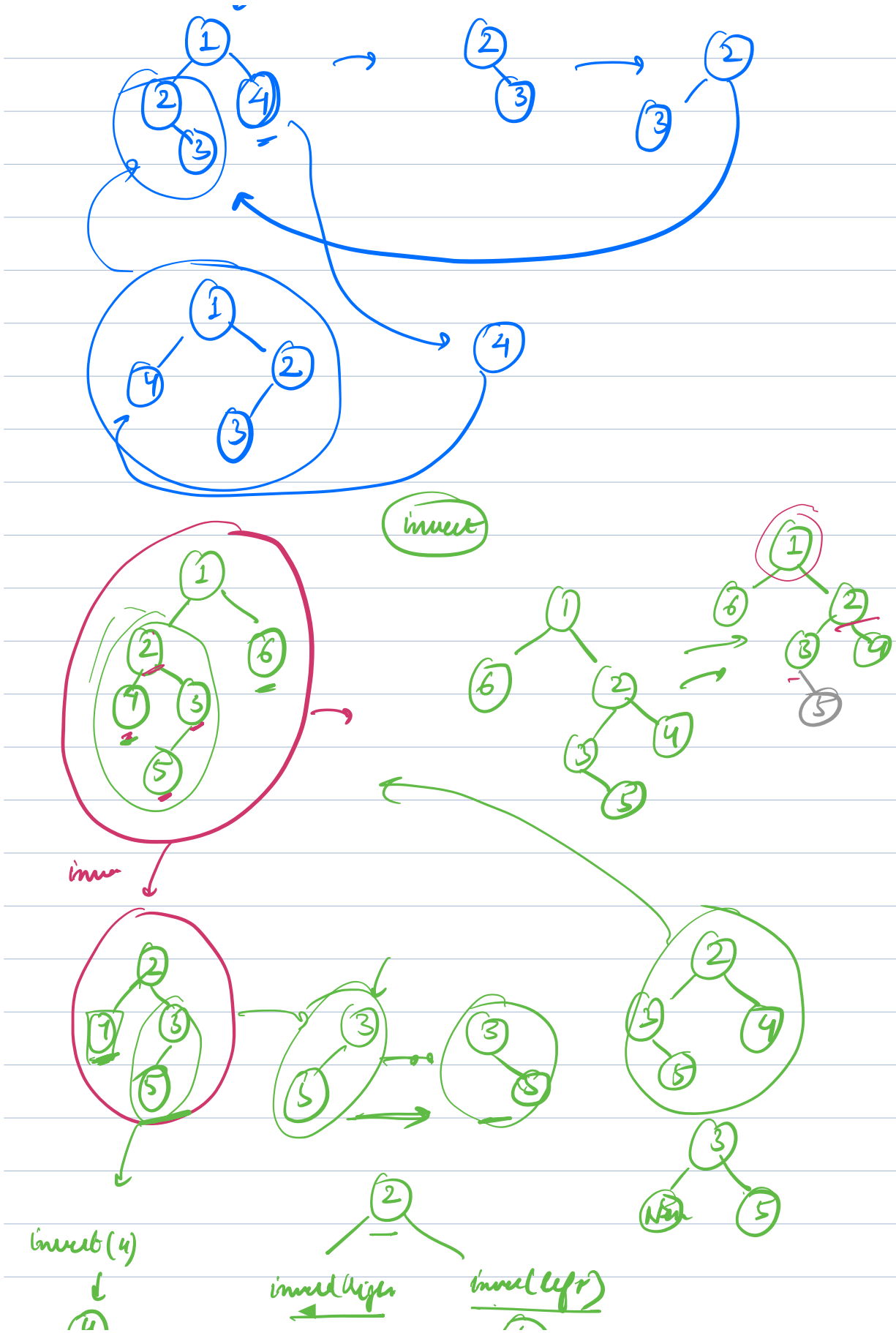






root.left = 0x

✓


invert
invert
invert(4)
invert(right)
invert(left)

(1) (3) (4)

3
5

Binary Search Tree | BST

7 mins

No duplicates

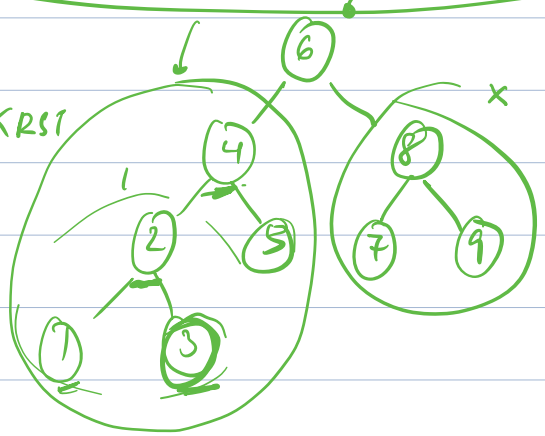
Root

LST RST

All nodes in LST < Root.val

All nodes in RST > Root.val

BST ?



LST < Root < RST

search 3

Searching is V fast

We need to look at all nodes

Inorder Traversal of BST :

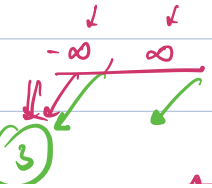
1, 2, 3, 4, 5, 6, 7, 8, 9

sorted?

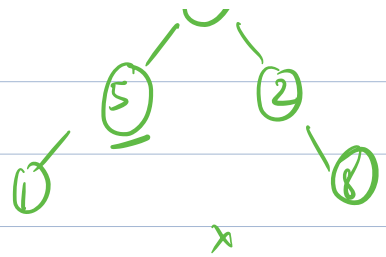
LST, Root, RST

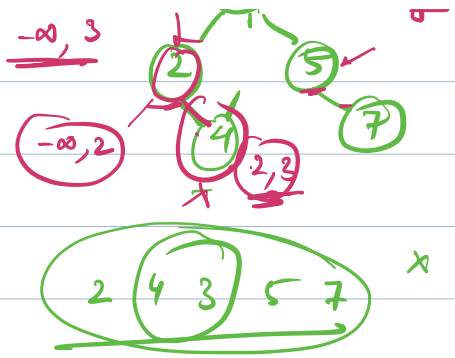
LST < Root < RST

(*) Inorder Traversal of a BST is sorted.



(3)





2 4 3 5 7

Q Given a binary tree. check if the tree is a BST or not. ?

→ Inorder Traversal of tree.

↓

O(N) → check if it is sorted → Yes

(root, Min, Max)

```
l = []
def inorder( root ) :
    if root == None
    return
    inorder ( root. left )
    l. appedd ( root. val )
    inorder ( root . right )
```

a[i] < a[i+1]

```
def checkBST ( root , max , min ):
```

is_left = check BST (root.left, root.val, min)

is_right = check BST (root.right, max, root.val)

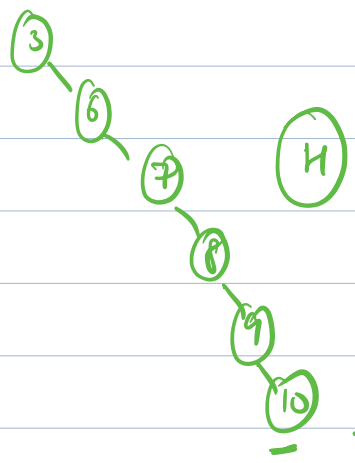
return is_left and is_right and

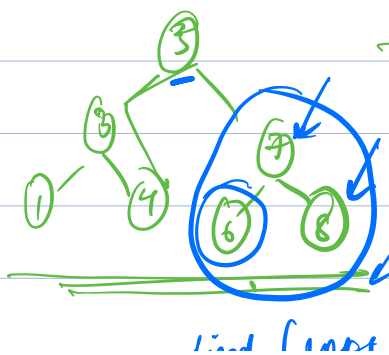
root.left.val < root.val

and root.right.val > root.val

Q Given a BST, find if value K is present in BST.

```
def find (root, k):
    if root == None:
        return False
    if root.val == k
        return True
    if k > root.val:
        return find (root.right, k)
    else
        return find (root.left, k)
```





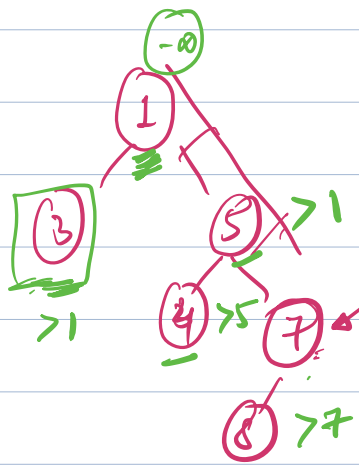
O(H)

BBST ⇒ O(LogN)

find (root, k)

Q2 count nodes with more value than all its ancs

Value > max(ancestors)



>1

counting nodes in the tree rooted at root with val > anc_max

```
def count(root, ancestor_max):
    if root == None:
        return false
    cnt1 = count(root.left, max(anc_max, root.val))
    cnt2 = count(root.right, max(anc_max, root.val))
    return cnt1 + cnt2 + root.val > anc_max
```