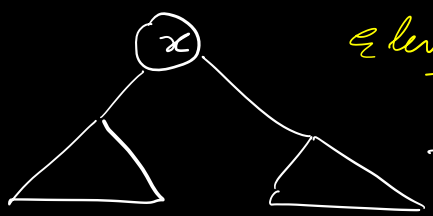
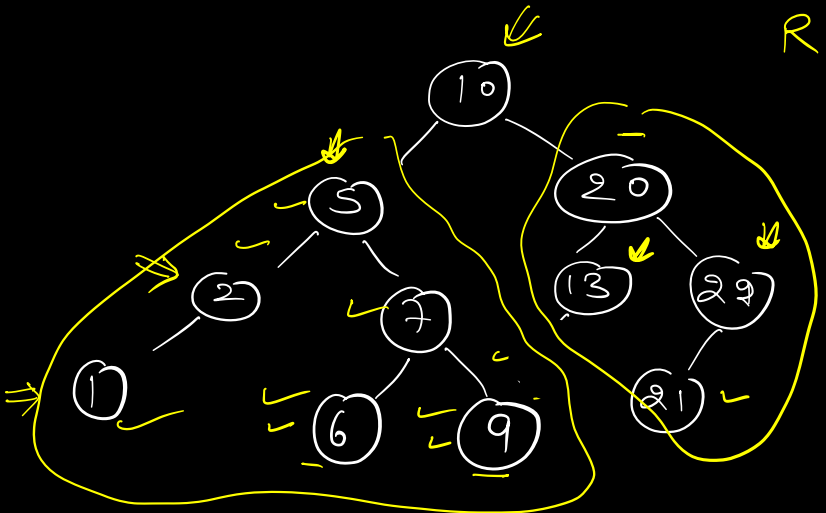
Binary Search Tree

1) It is a binary tree.

2)





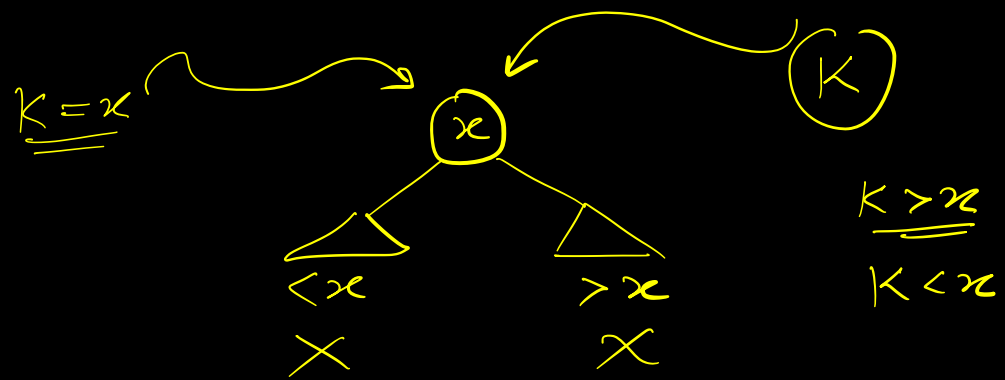
Q ⇒ How to handle case of equality

1) < value, freq > as node

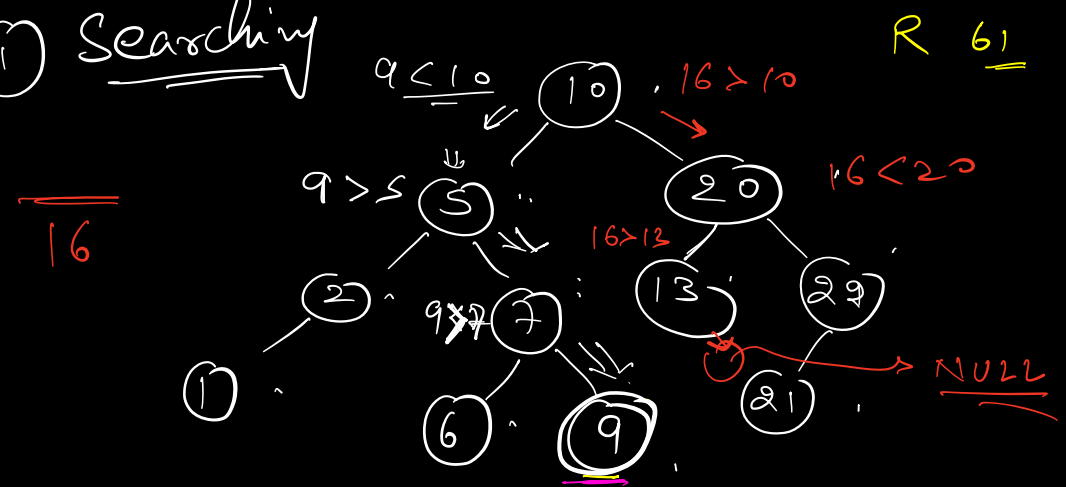
2) Either always in the left or always in the right.

Q We have to search for an element K in the BST??

1) Traverse across all nodes. (O(N))



① Searching



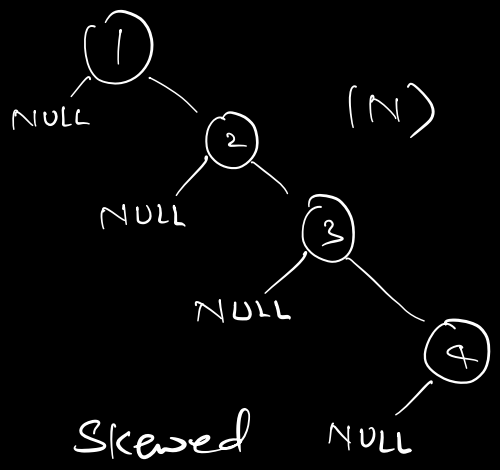
R 61

9

T.C. = O(H) < O(N) ⇒ worst case, O(log N) ⇒ Best case

Q ⇒ When to stop searching

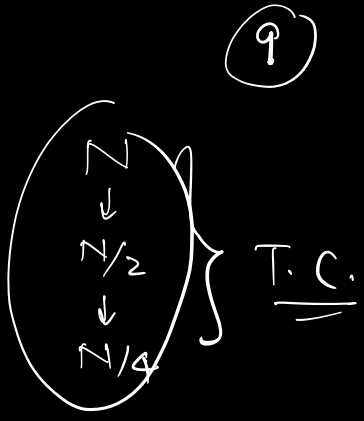
⇒ When we reach a NULL



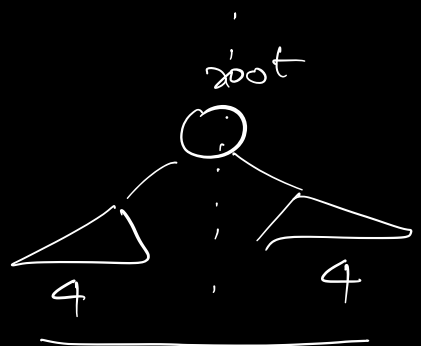
(N)

$H = O(N)$

Skewed Tree



T.C.

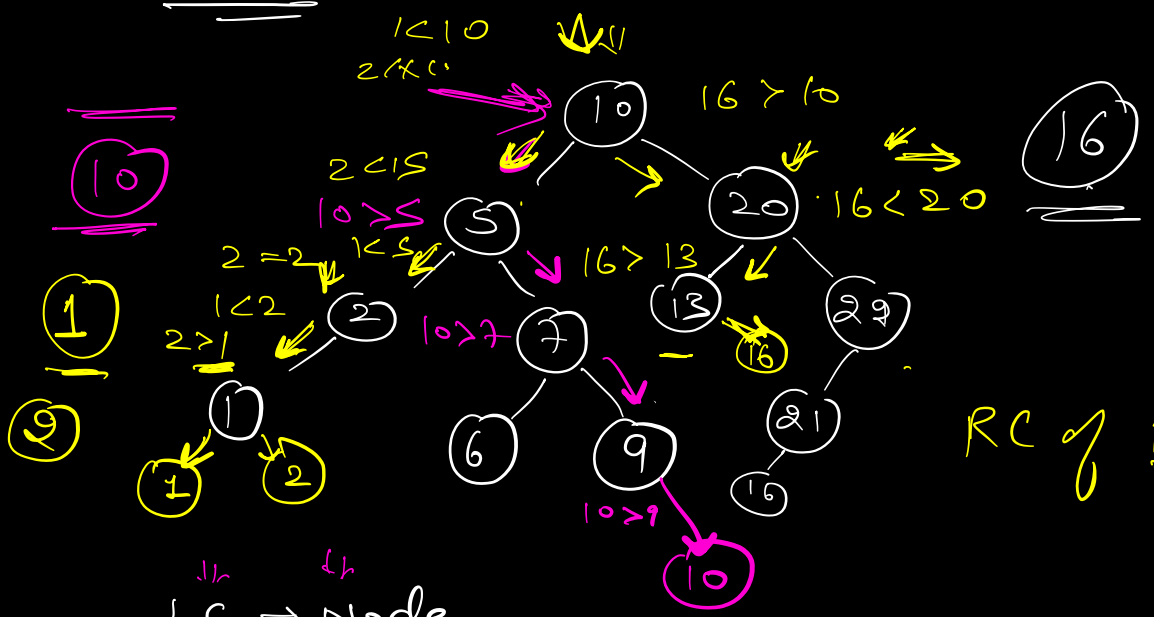


Balanced Tree

$O(\log(N))$

② Insert

Equal elements in LST / RST



RC of 13

LC ⇒ Node

RC ⇒ Node

1) If 16 was present, where would it be present. (Search for 16)

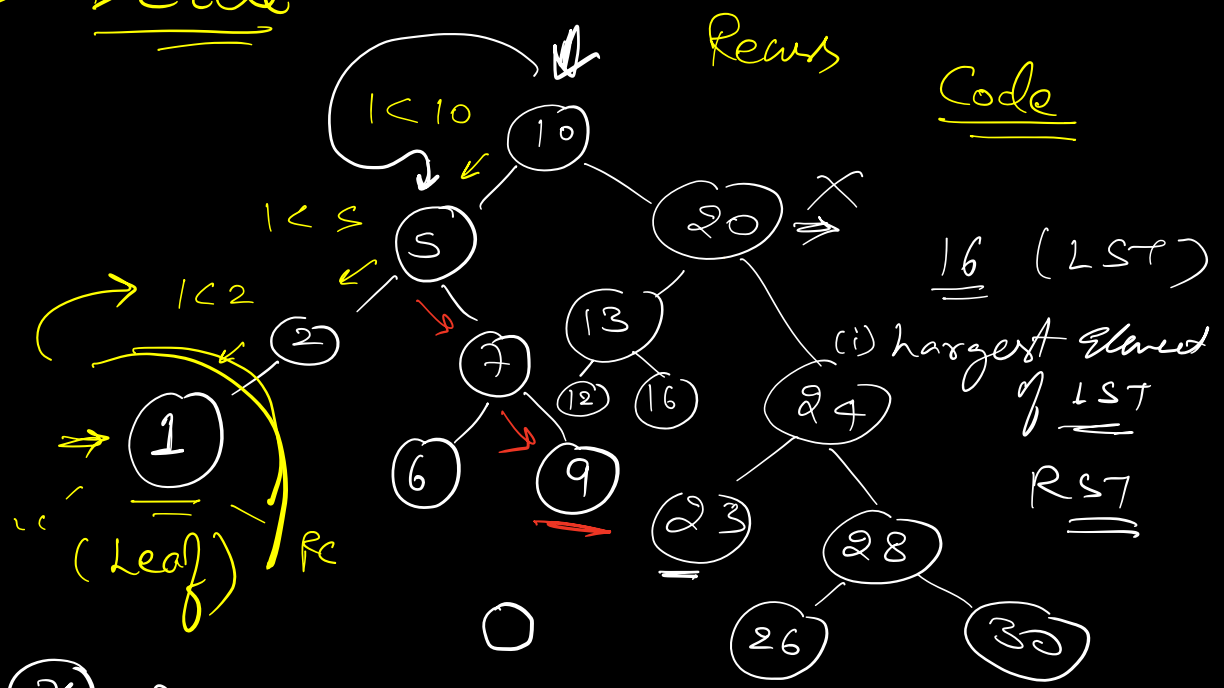
2) Insert 16 at that position ⇒ O(1)

parent → right = new Node(x)

T.C. = O(H) + O(1)

= O(H)

(3) Delete



(x) RST > x

1) Delete node (1) (leaf)

(i) Search for the node

(ii) Delete the leaf node

(2.left = NULL)

1 = NULL

2) Delete the Node (25)

(Node with 1 child)

⇒ Replace the node to be deleted with the child.

3) Node with 2 child

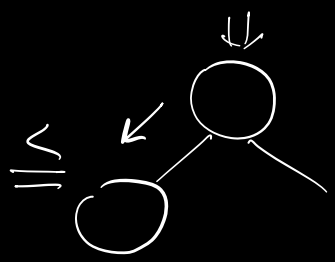
(i) a) largest from the left subtree

```
node = node.left;
while (node.right != NULL){
    node = node.right;
}
return node;
```

b) Replace the node to delete with the Largest of LST.

c) Delete the node.

(ii) Smallest Element of RST



Case 3

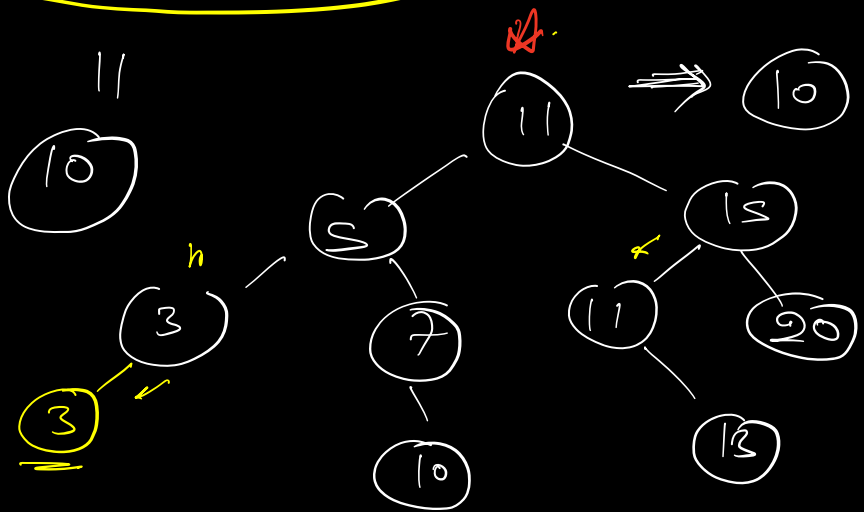
OR

greatest of the left subtree

(Inorder predecessor)

Smallest of the Right subtree

(Inorder Successor)



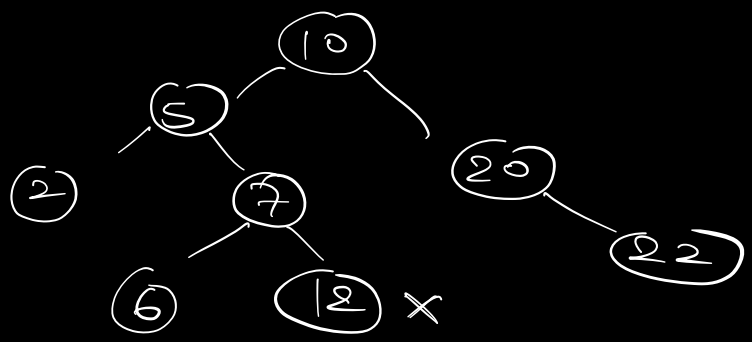
LS ⇒ Inorder predecessor

RS ⇒ " Successor

NO

CODE ⇒ End

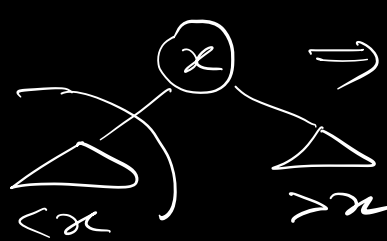
Q Given a Binary Tree
Check if the BT is a BST.

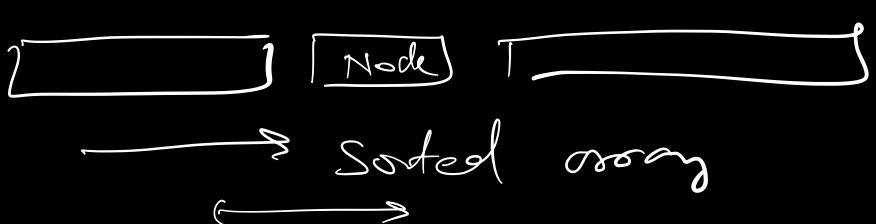


Soln

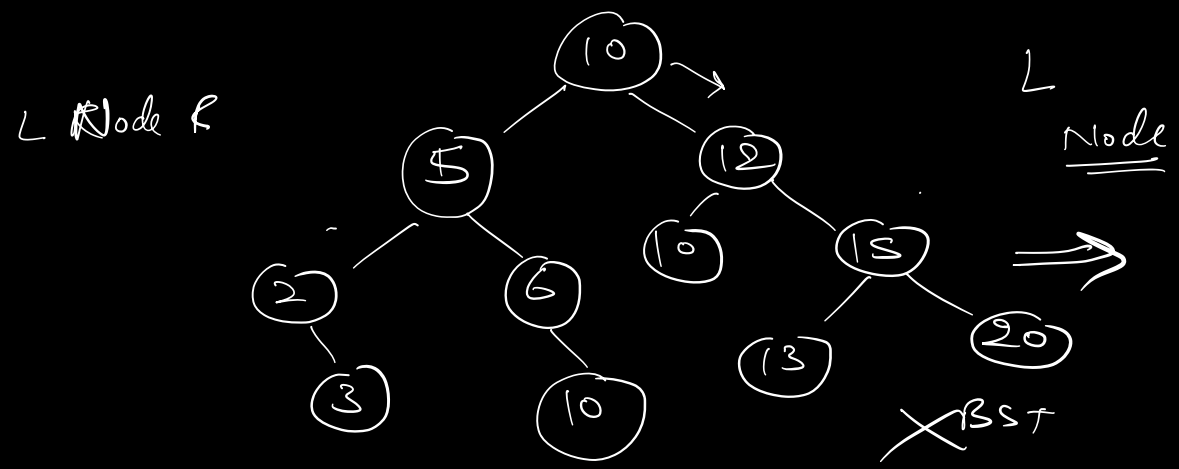
(1) Pre — Node, Left, Right
In = Left, Node, Right
Post = Left, Right, Node.

BST





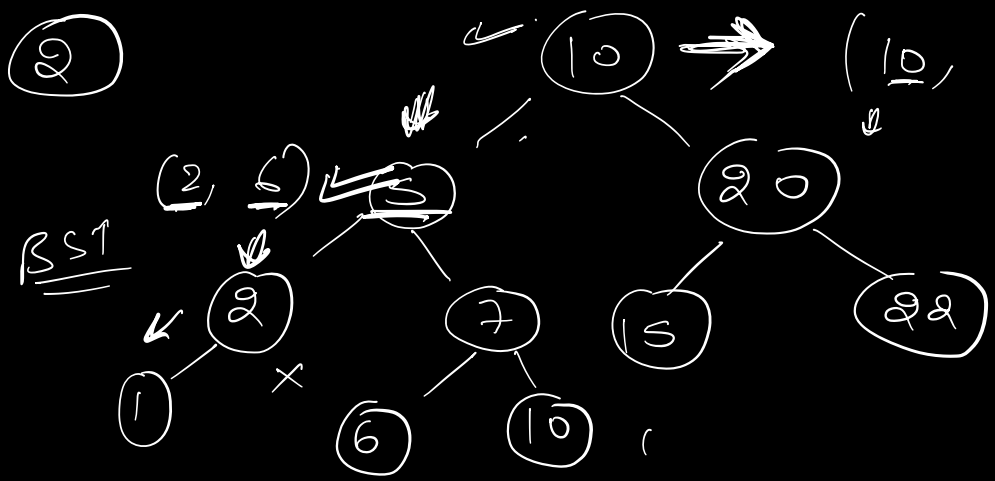
⇒ If inorder traversal is sorted, the the tree is a binary search tree.



⇓

2, 3, 5, 6, 10, 10, 10, 12, 13, 15, 20

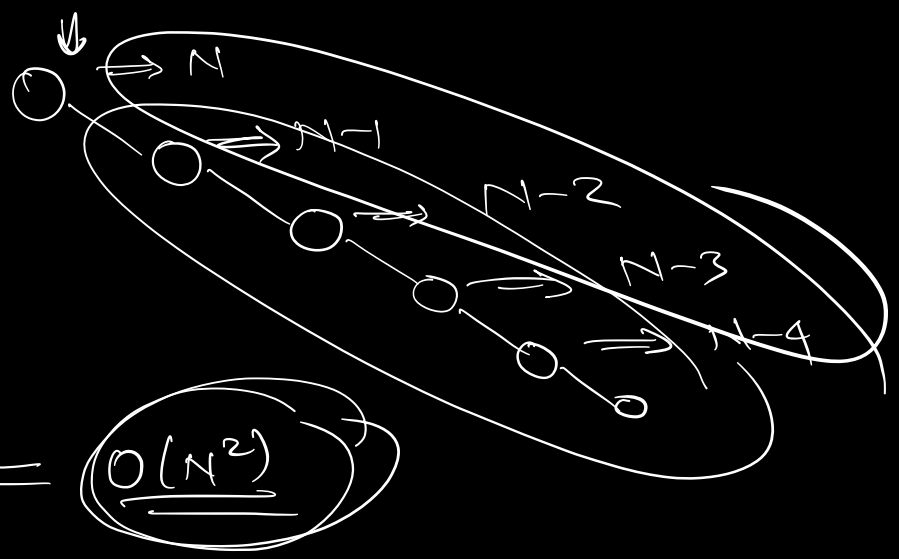
Node, left, right ⇒ Pre = (L S)



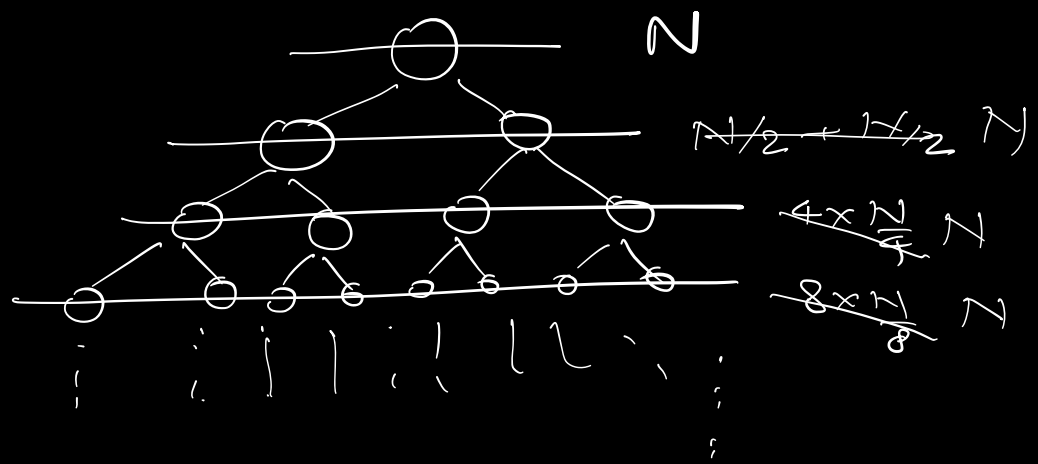
⇒ Max of the LST

$O(N + 2(\frac{N}{2}) + (\frac{N}{4})(\frac{N}{4})(\frac{N}{4})\ldots) = O(N \log N)$

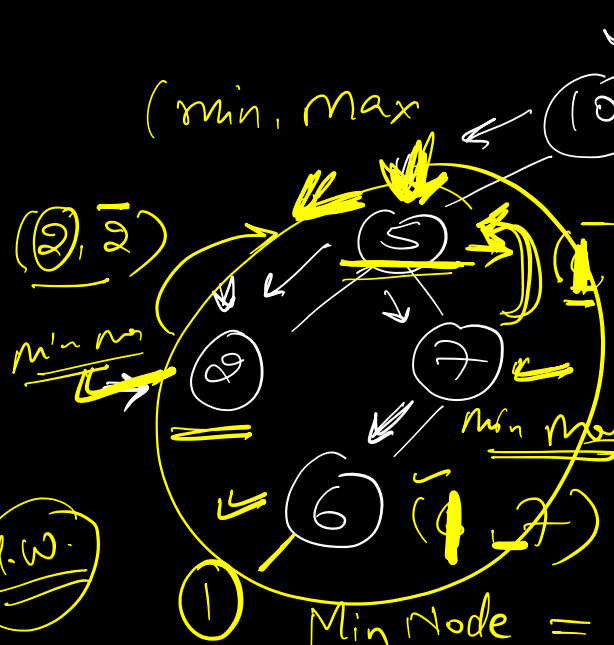
⇒ We need to check the entire left subtree & the entire right subtree.



$= O(N^2)$



⇒ Left, Right, Node ⇒ POST



L, R, N

(min, max)

(Min, Max)

NO X

H.W.

MinNode = min (minLeft, Node.value, minRight)

PRE ↓

POST ↑

max left < 5 < min right