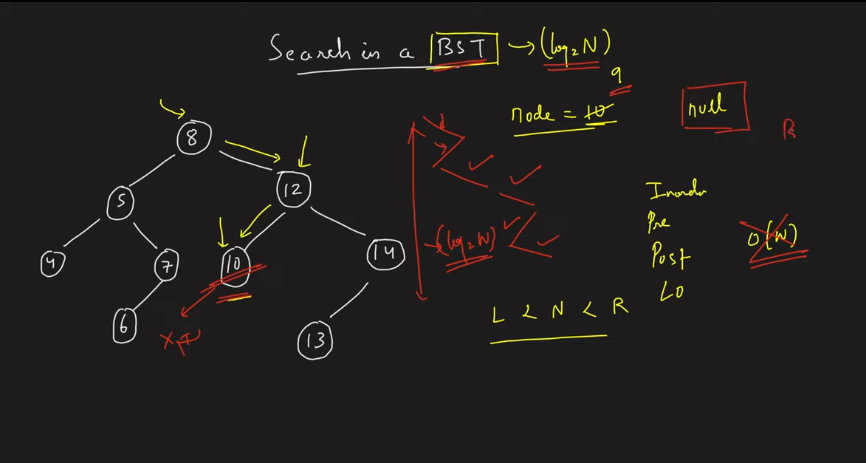
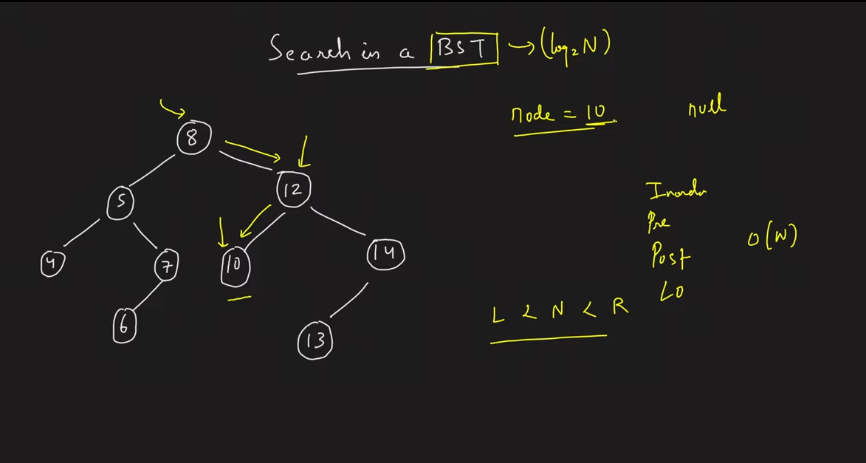
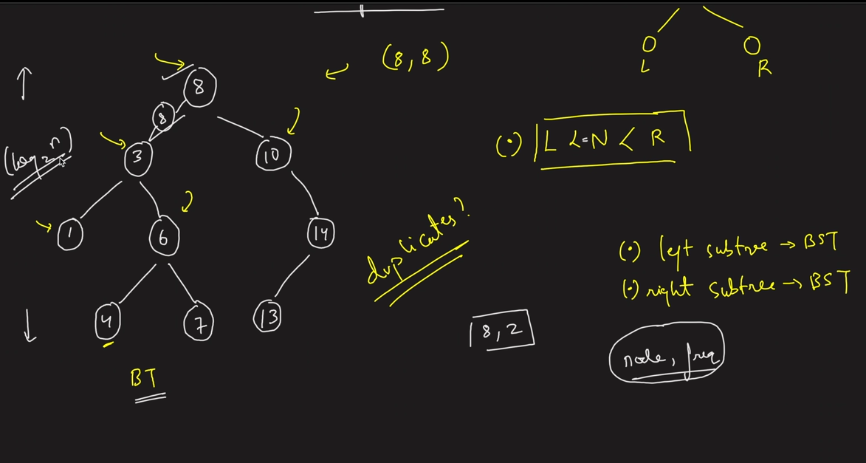
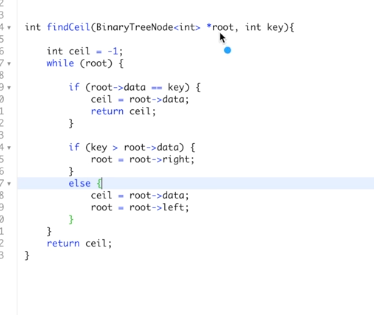
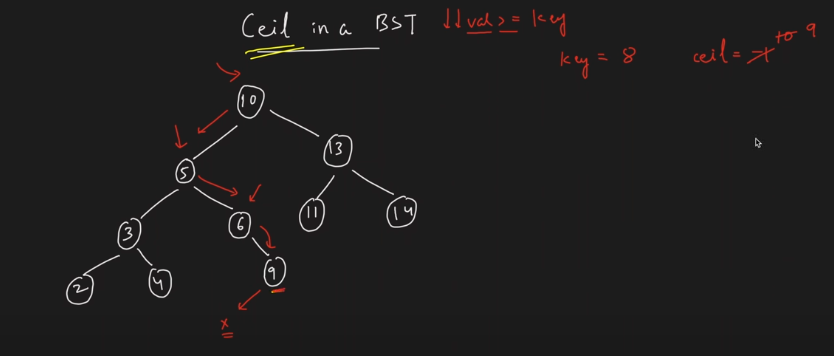
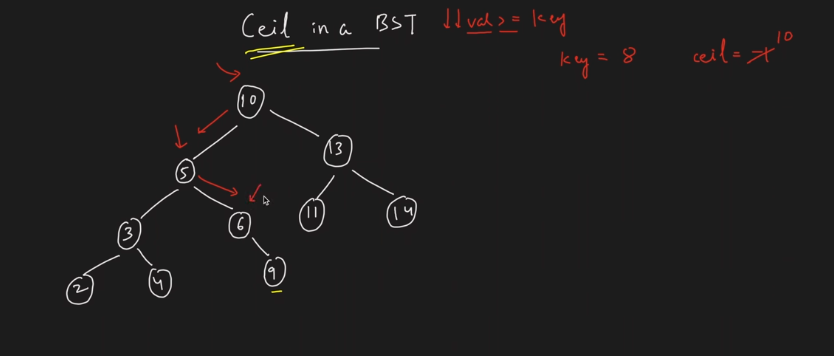
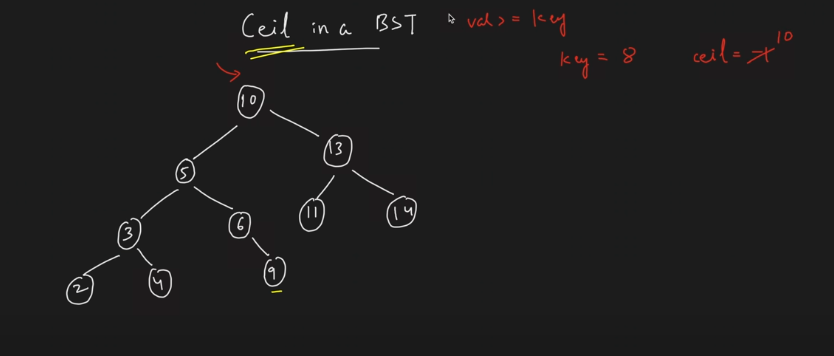
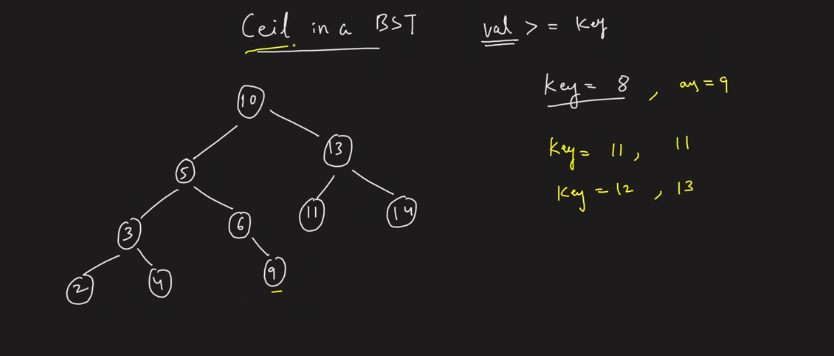


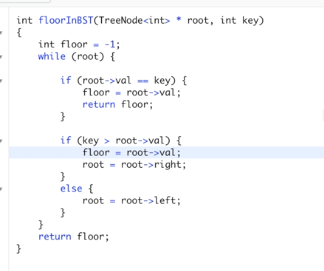
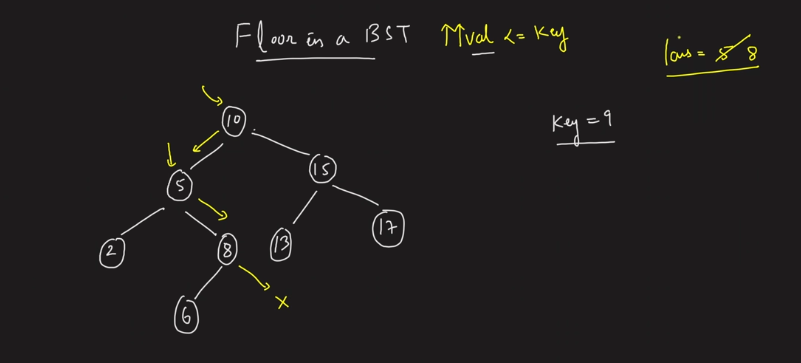
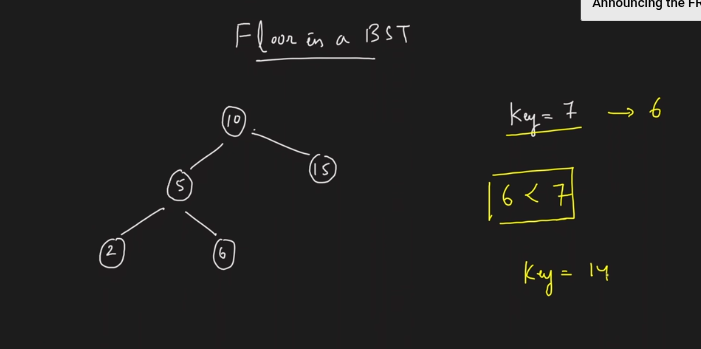
If duplicates



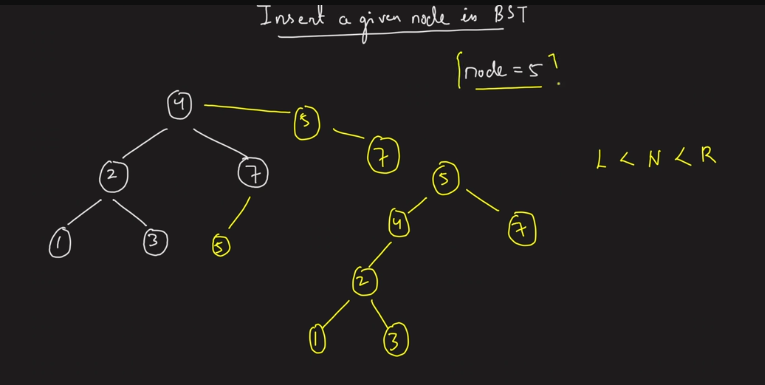
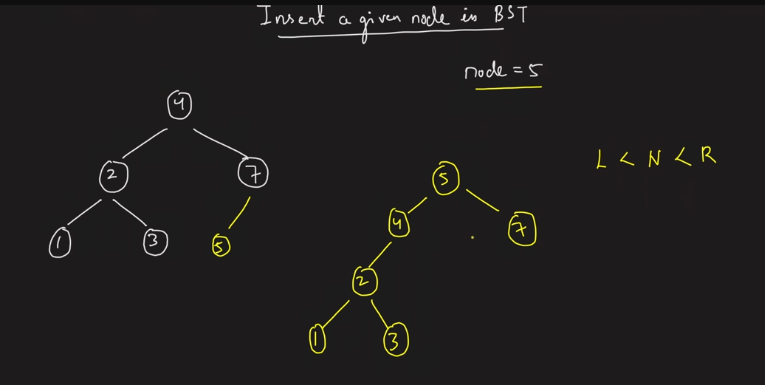
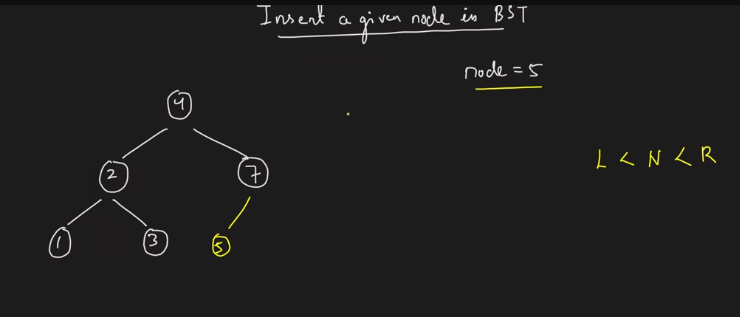
CEIL



**Floor**

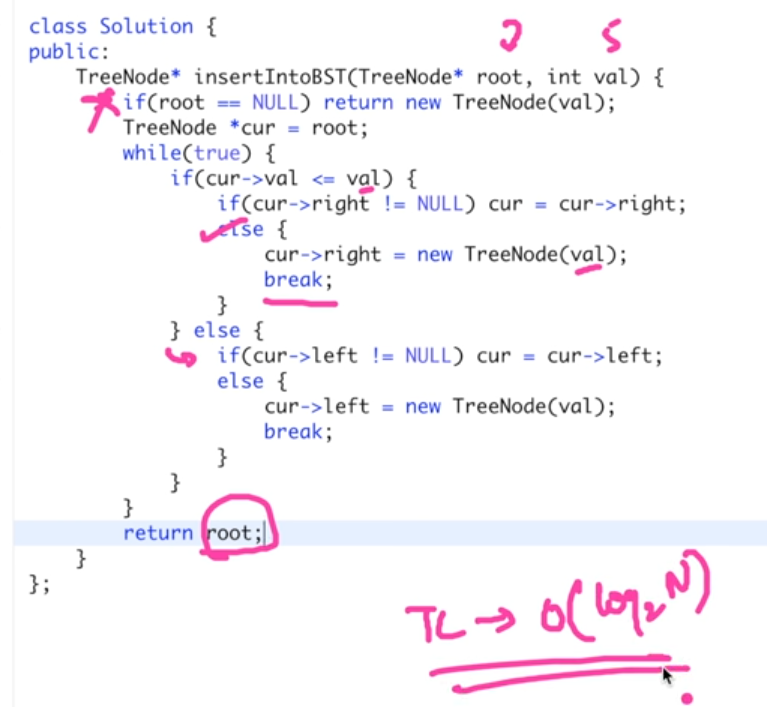
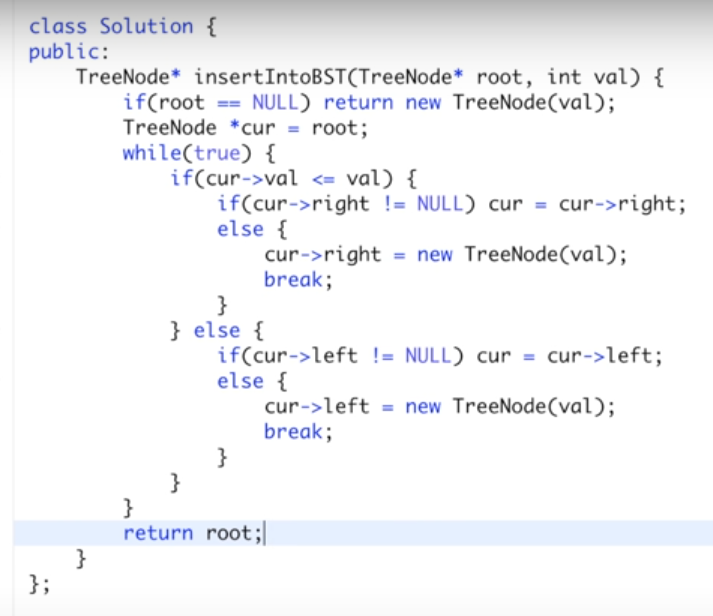
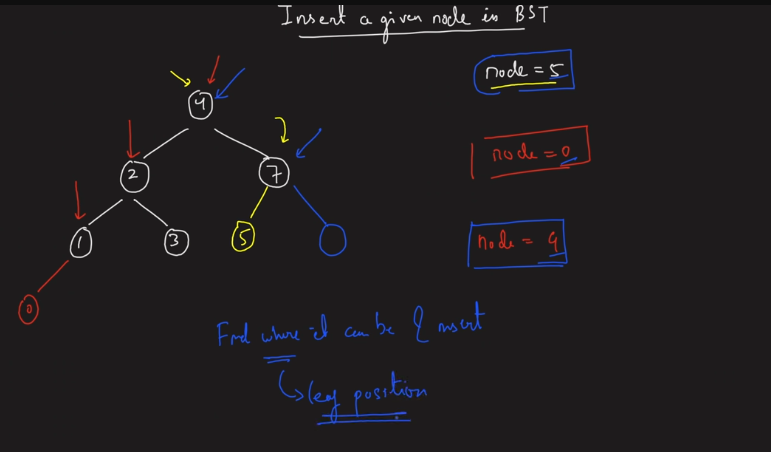
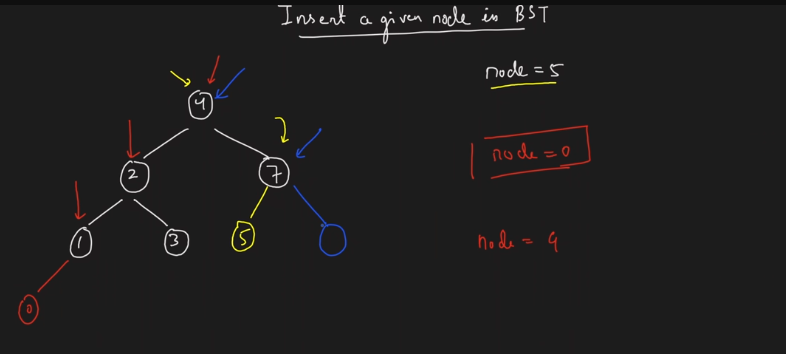
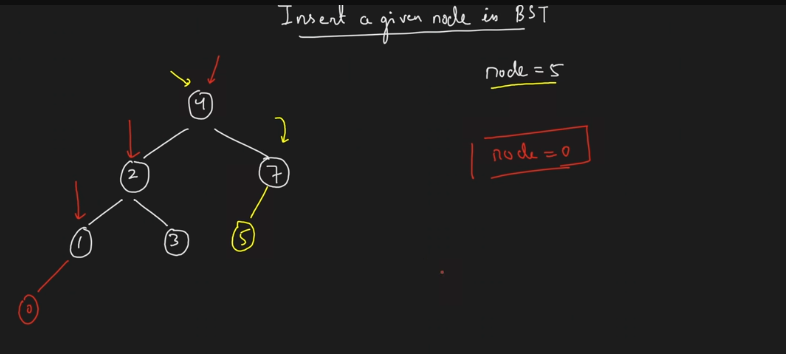
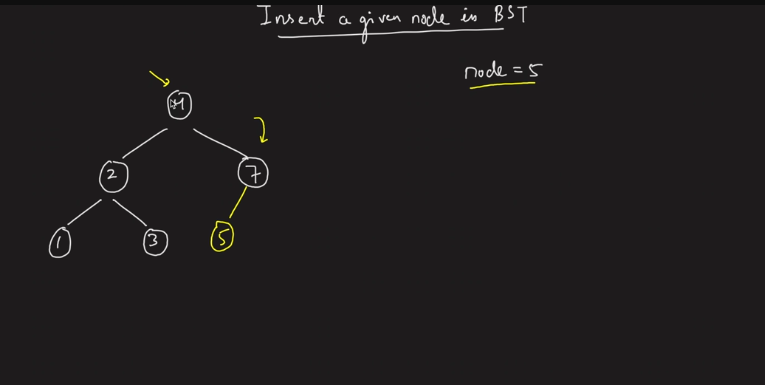


INSERT A GIVEN NODE IN BST

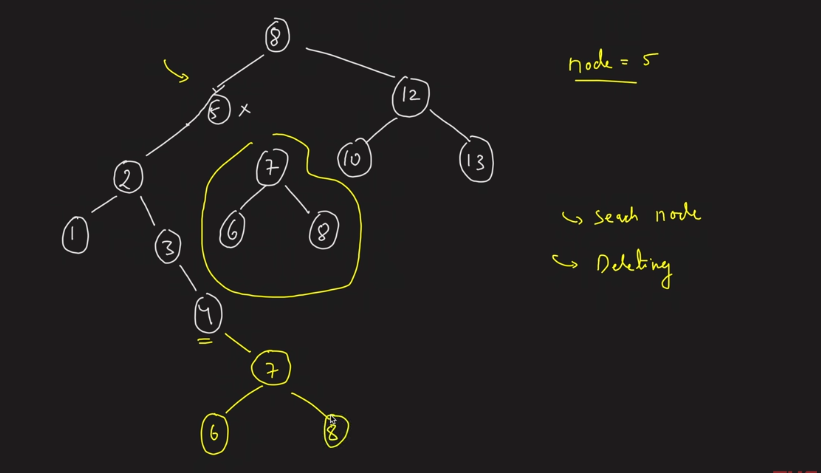
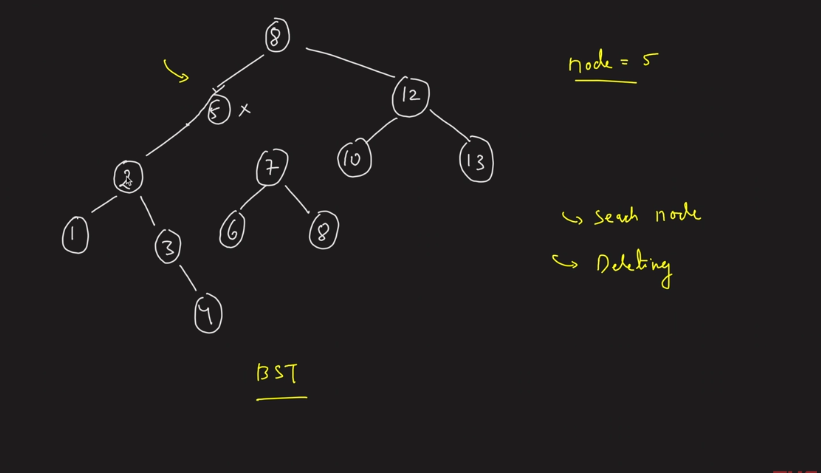
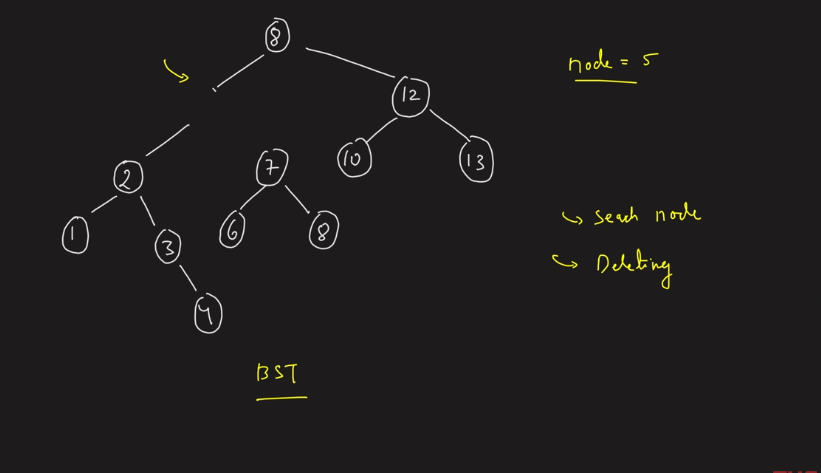
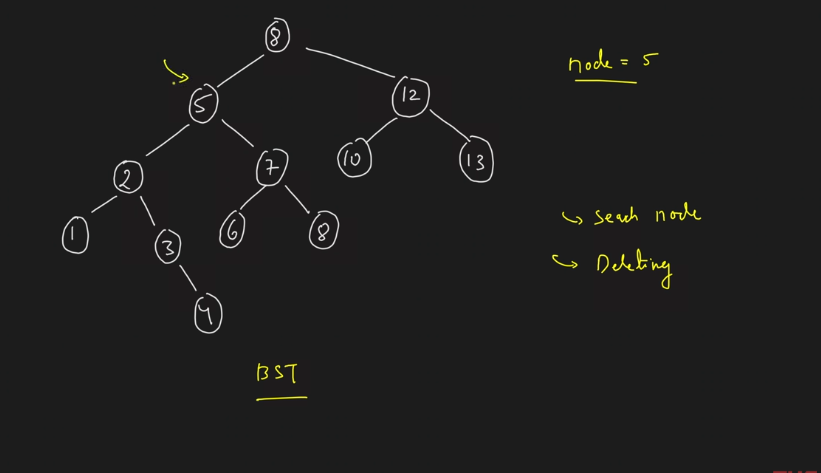
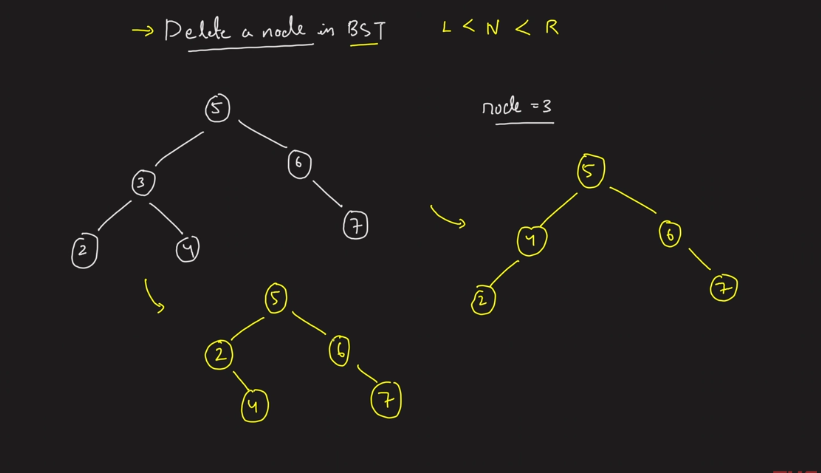


Need to find an one ans return the answer

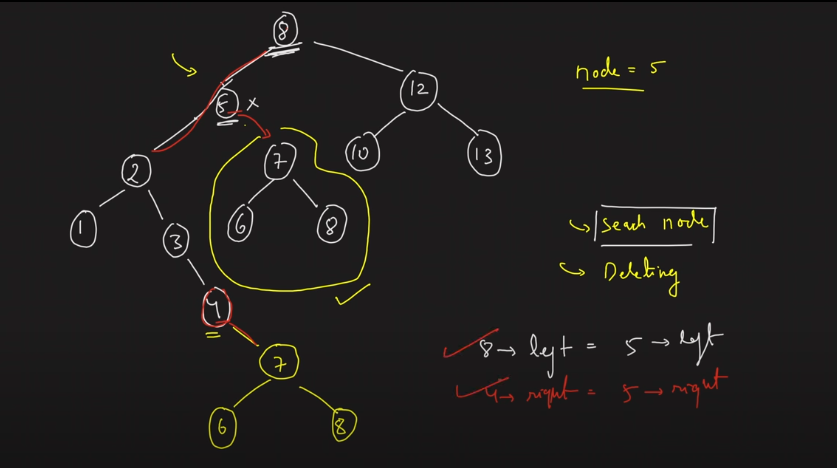
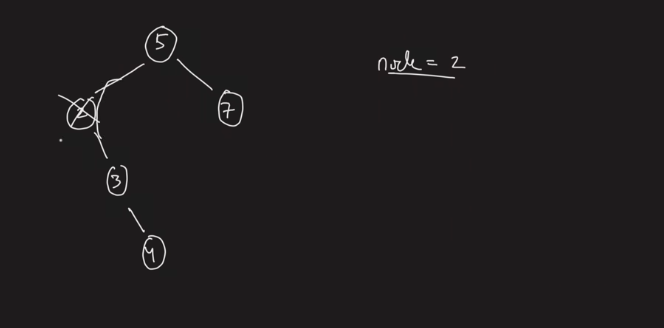
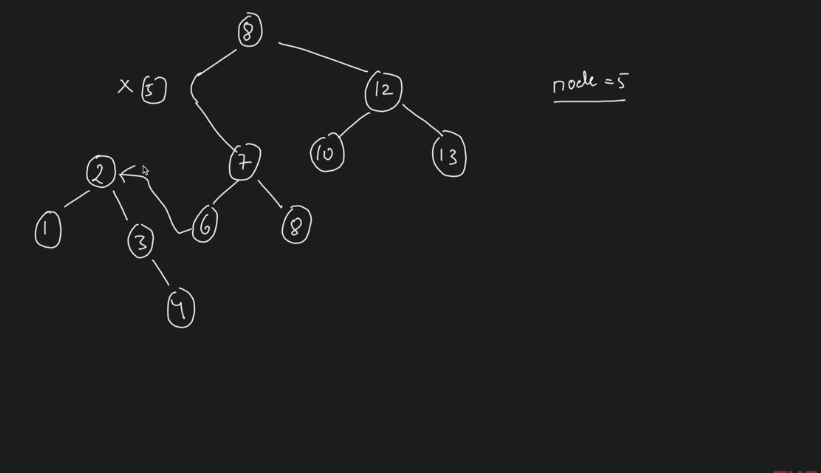
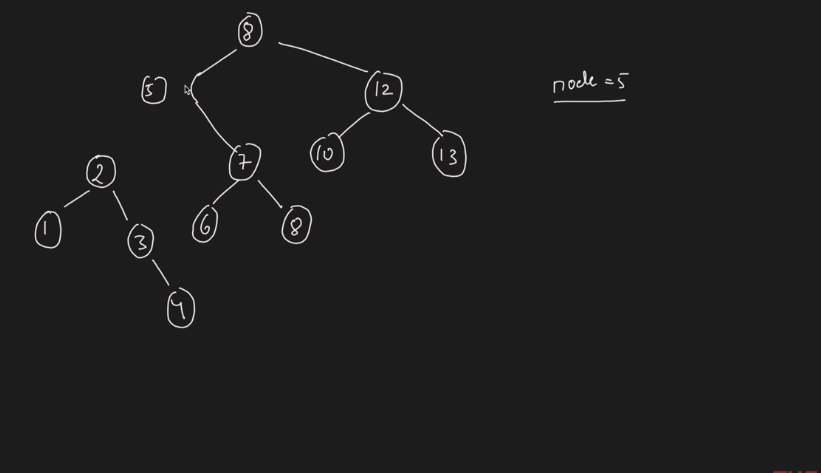
1. Leaf insertion



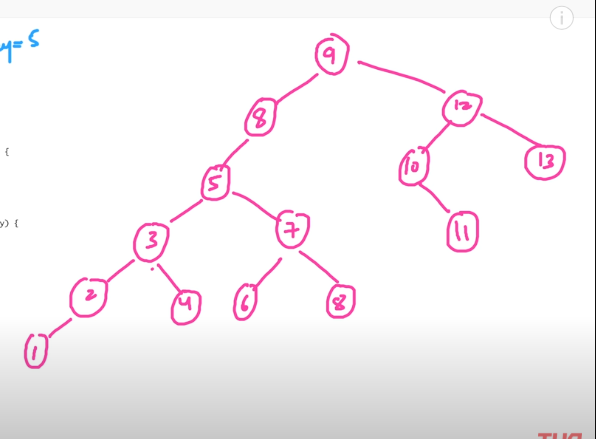
**DELETE A NODE**



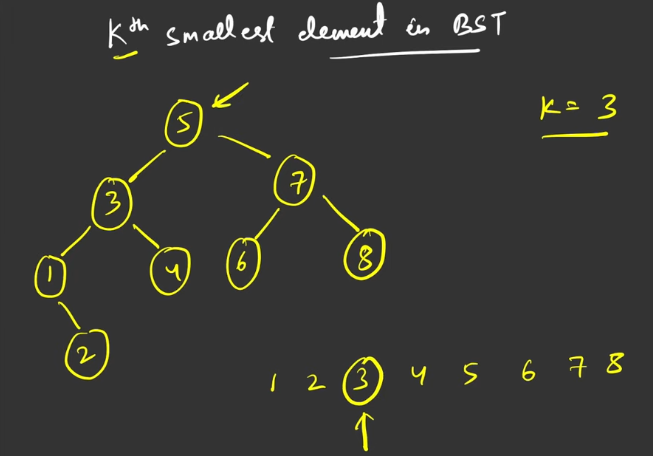
2nd way



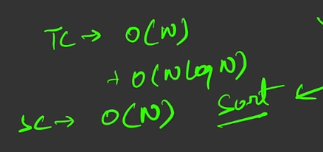
Trace the above for the below graph



KTH SMALLEST ELEMENT IN BST

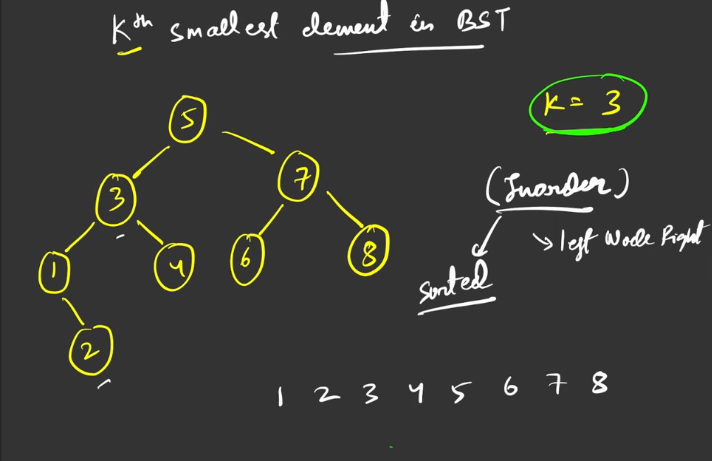


Naive: BFS -> STORE NODE-> VECTOR LIST-> SORT-> RETURN KTH



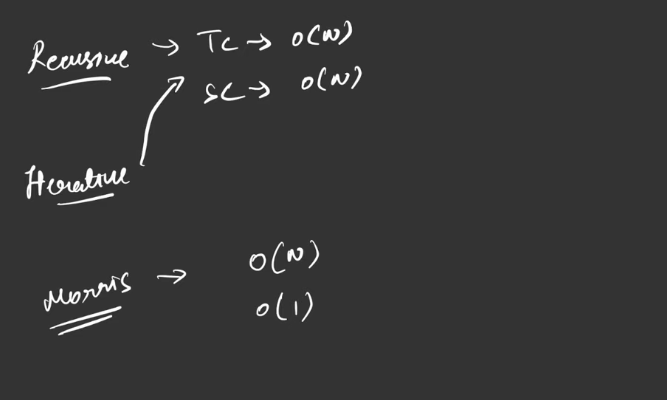
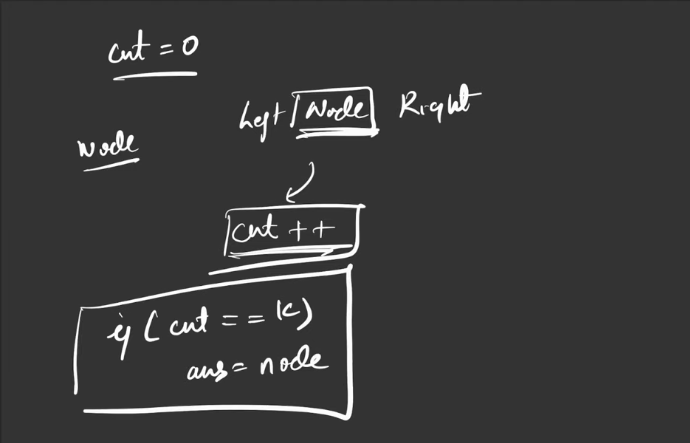
Optimised version:

Inorder of the bst – sorted order.

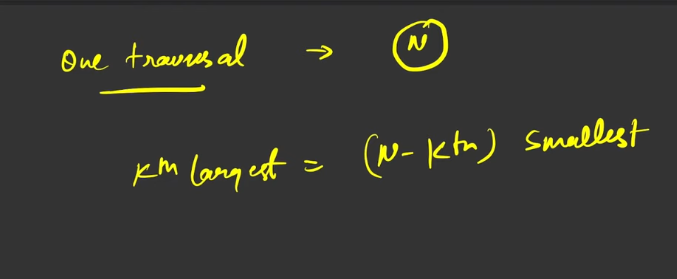


Storing the inoder and then giving answer: Tc=O(N), SC=O(N)

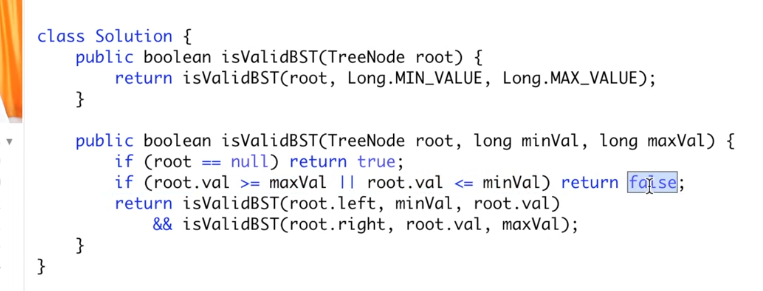
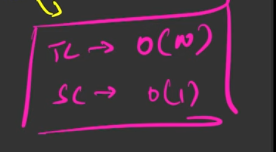
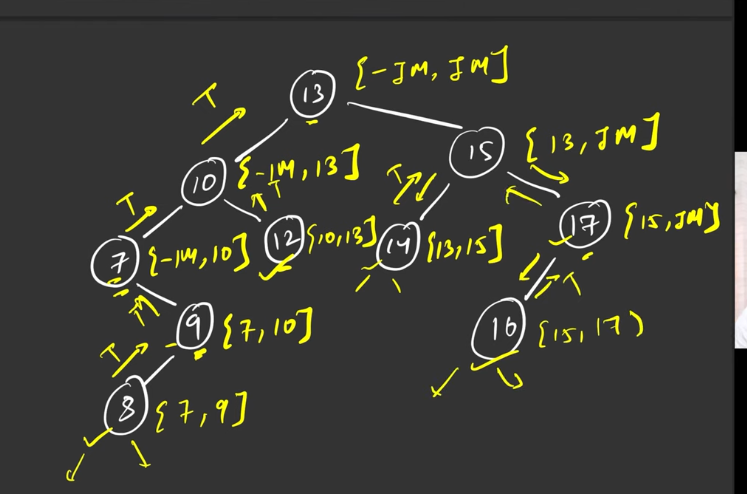
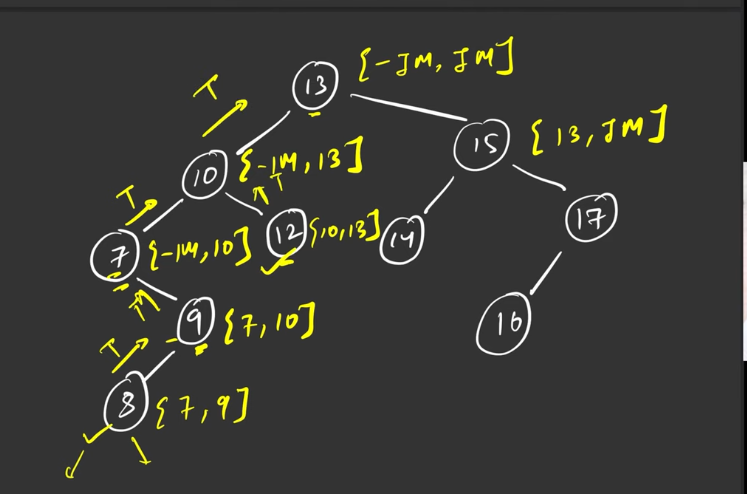
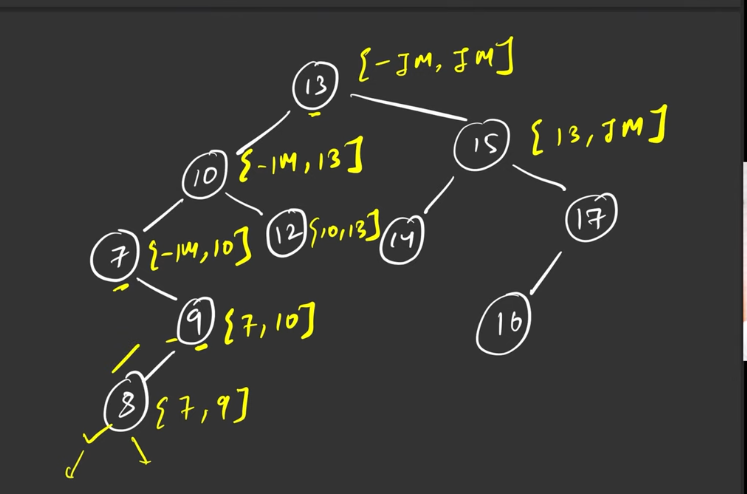
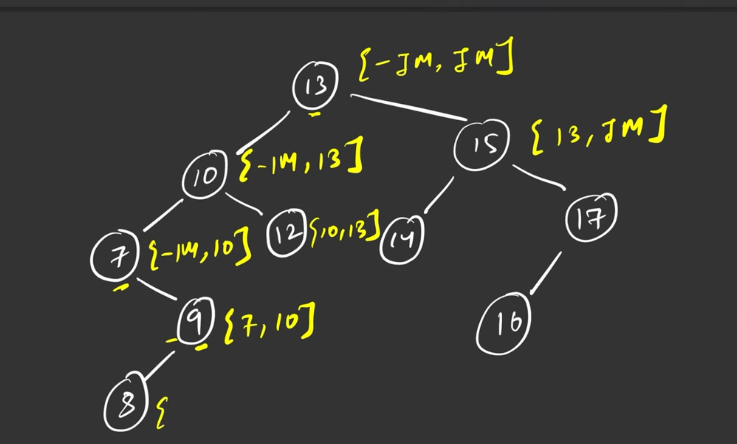
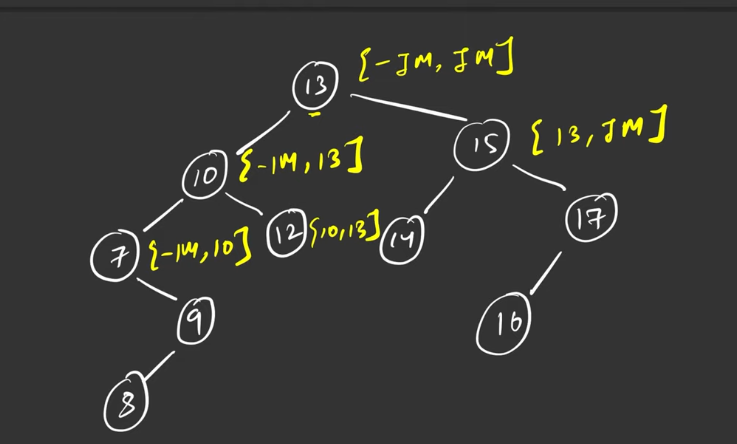
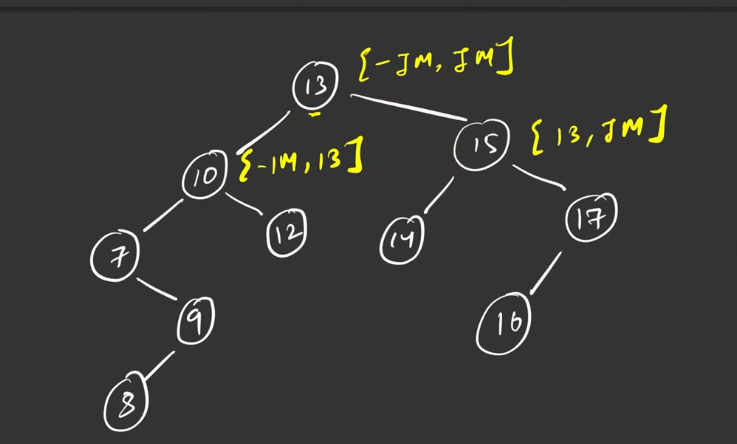
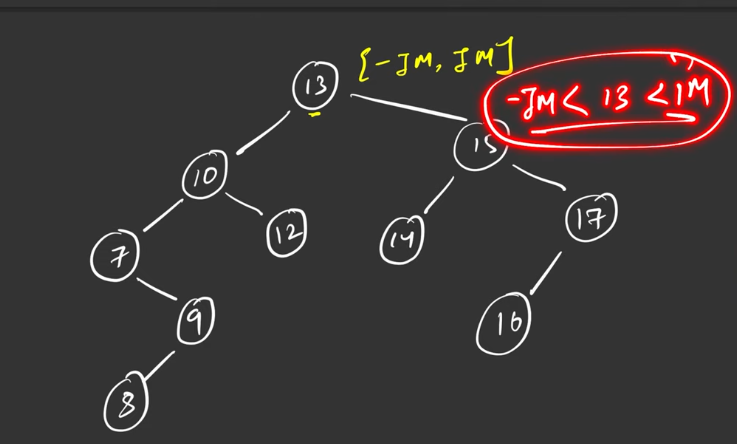
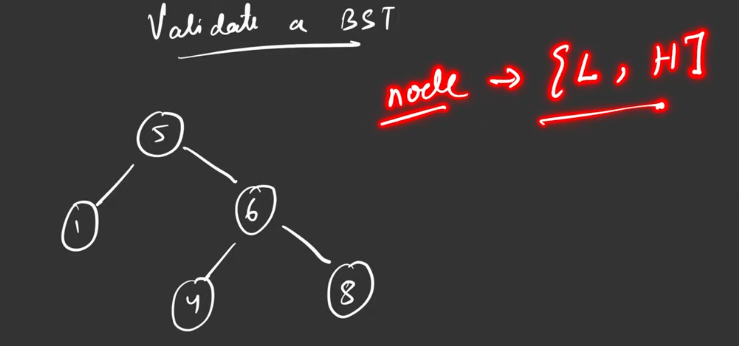
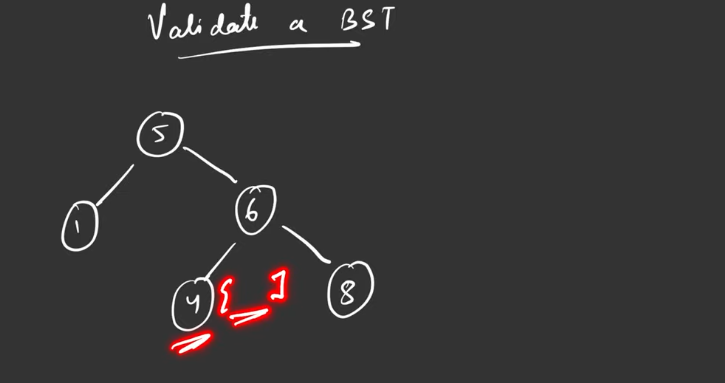
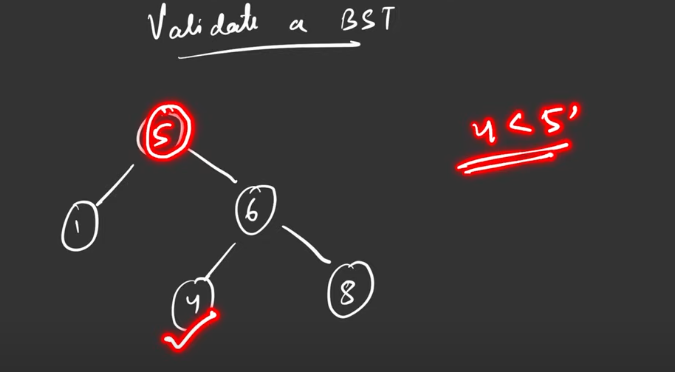
Still optimised with the help of the count



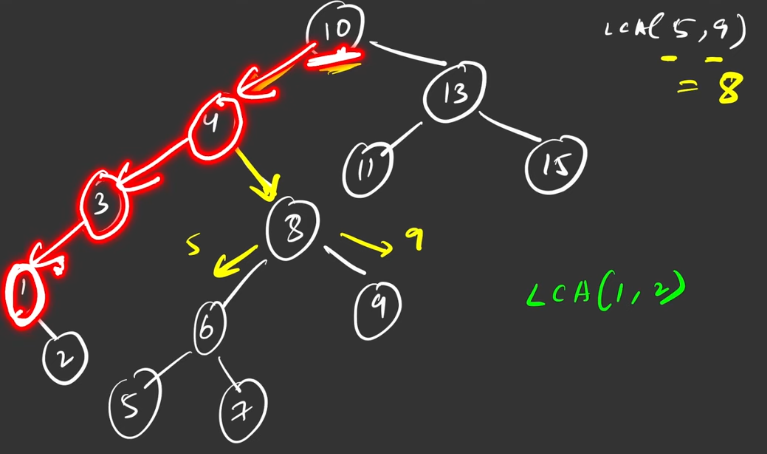
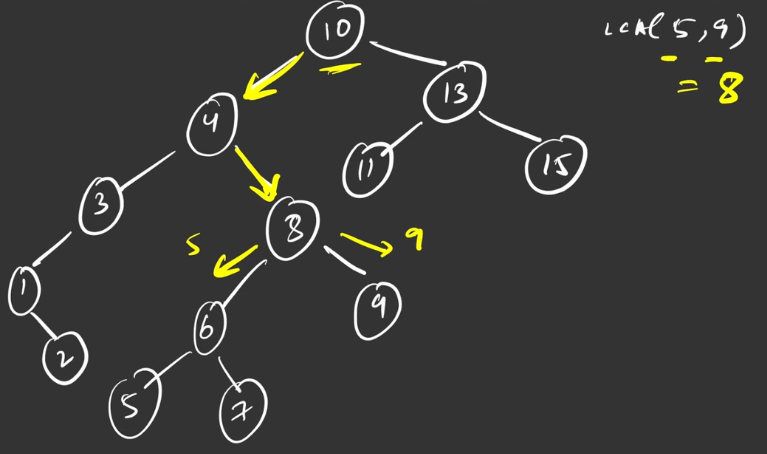
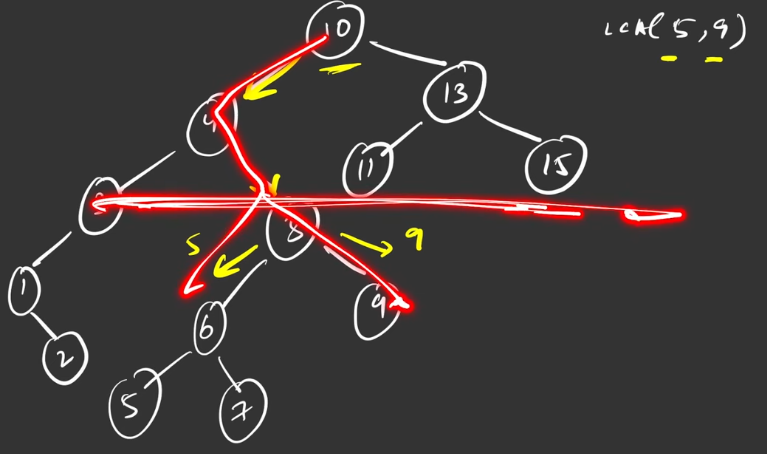
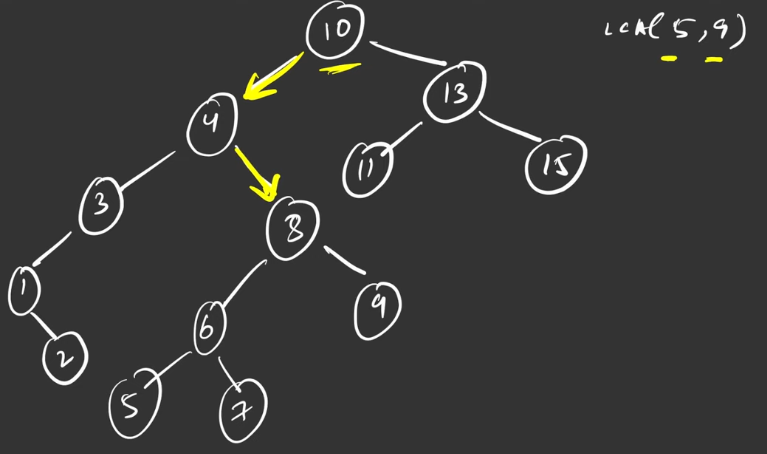
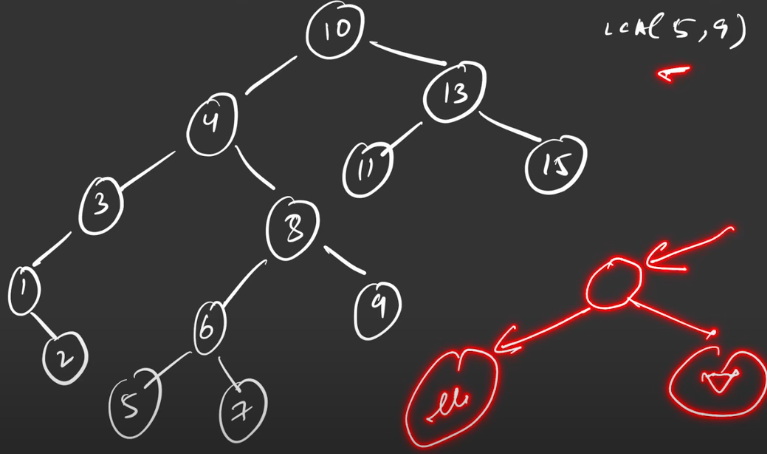
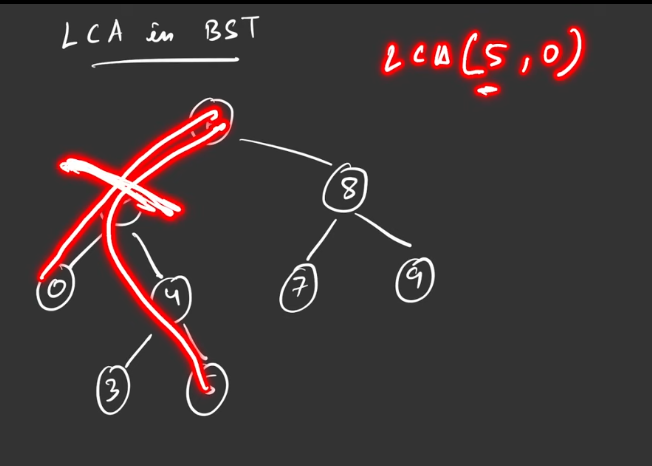
**K- Th largest:**



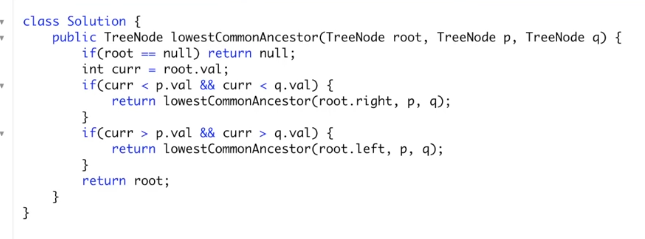
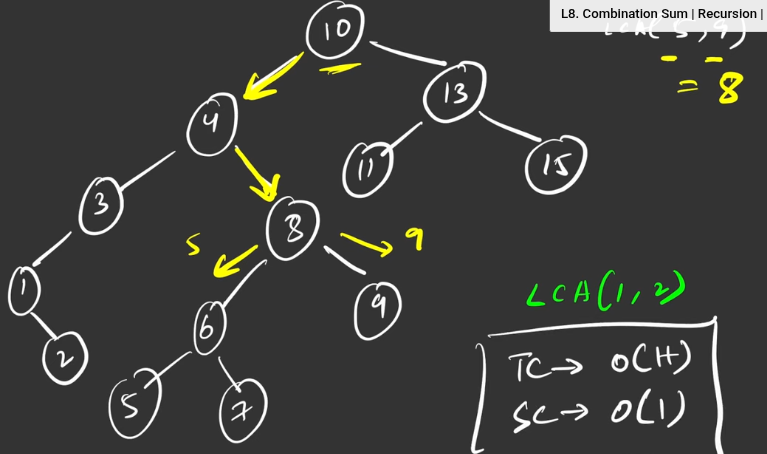
**VALIDATE BST**



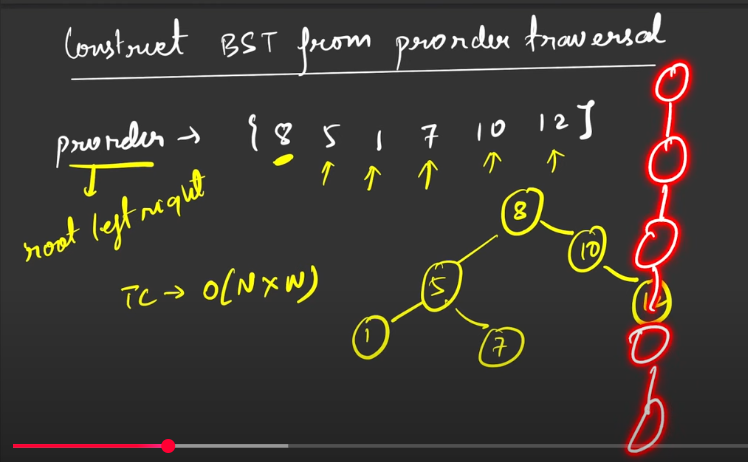
**LCA IN BST**



If both of hem are on the right , hence we mve to right. If both of them are on tge lesft move to left. If not then that is the lca



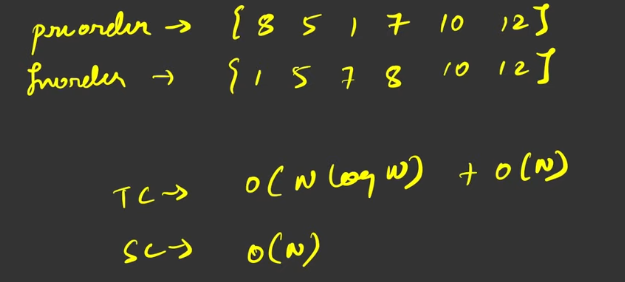
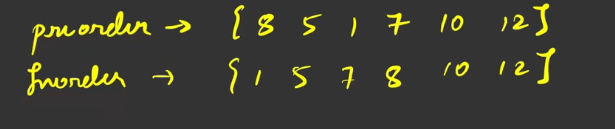
CONSTRUCT BST FROM PREORDER.



To traverse all the nodes it ha to take the O(N), to decide where to keep it, to move to the posotion(especially when it is the skew tree) it also takes O(N), hence O(N^2).

**OPTIMIAL**:

1. sort preorder to get the inorder.



This is from the previous way of building the tree from the preorder and inorder.

This log(n) to be removed.

1. Optimised
   1. This is the logic we are going to use, to validate whether tree is binary tree or BST.
   3. s
2. s