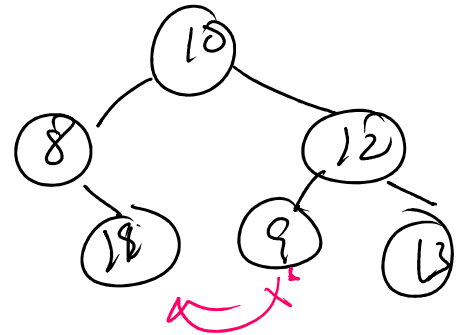
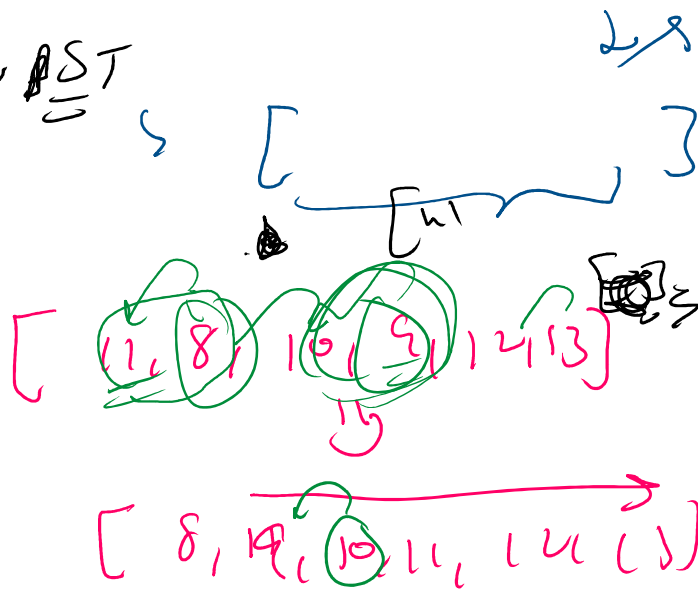


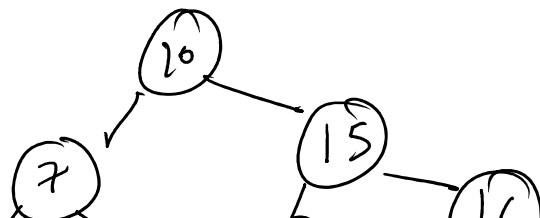
Serialize and Deserialize a BST

["process root"]
 $f(\text{root}.l)$
 $f(\text{root}.r)$

Recover BST



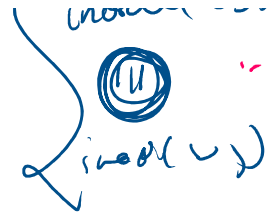
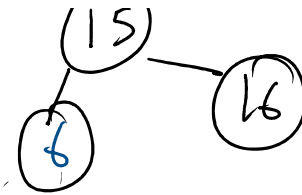
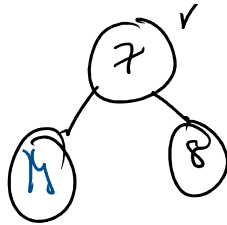
inorder (root, cur) {
 if (root == null) return;
 inorder (root.l, cur);
 // cur.add (root.val);
 inorder (root.r, cur);
}



(inorder (es))
 [11]

11, 7, 8, 10, 15, 16

arr[] >= prev[]
//
↓

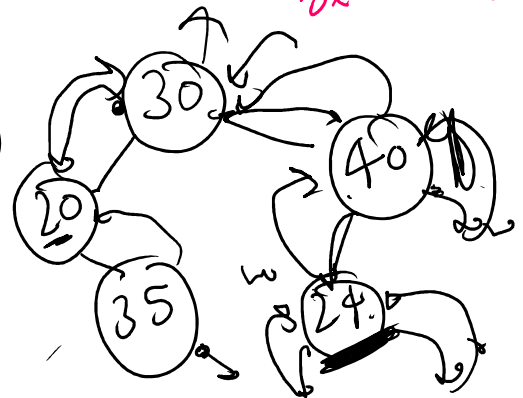


if (fd == null || prev > root.val)
fd = prev

if (fd != null && prev > root.val)
2nd = root

```

if(root==null) return;
inorder(root.left);
if(firstDefect==null && prevNode.val > root.val) {
    firstDefect = prevNode;
}
if(firstDefect!=null && prevNode.val > root.val) {
    secondDefect = root;
}
prevNode = root;
inorder(root.right);
  
```

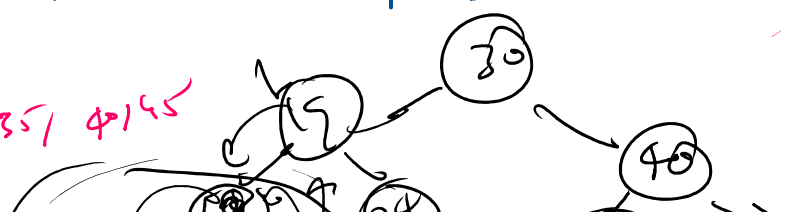


① It's not recommended to swap
addresses generally.

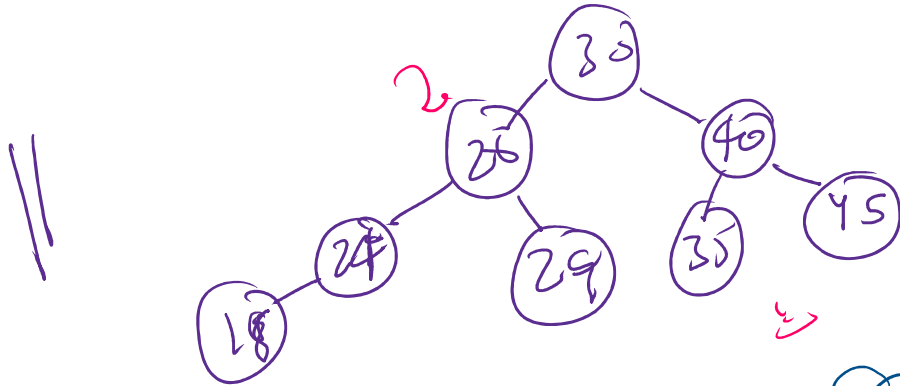
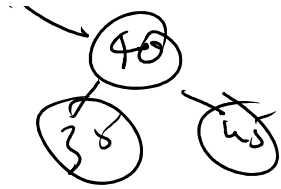
Successor: Turn here back
predecessor: Turn here



30, 35, 40, 45



16, 18, 19, 20, 24, 30, 35, 40, 45



18 24 26 28 29 30 35 40 45



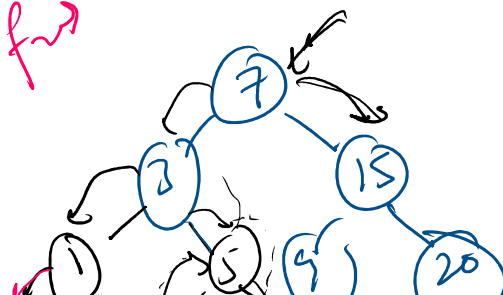
BST iterator

Stack



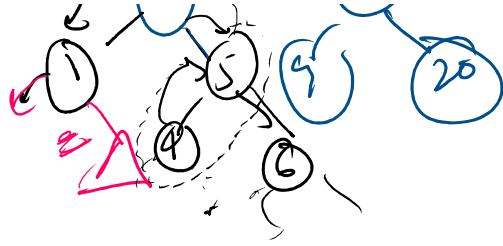
1

for



next()

Stack



1 3 4 5 6 7 9
20

Q ← insert
//

ⓑ Avg height of a BST (Tree is balanced)

$[O(\log n) \text{ to } \log_2 n]$

