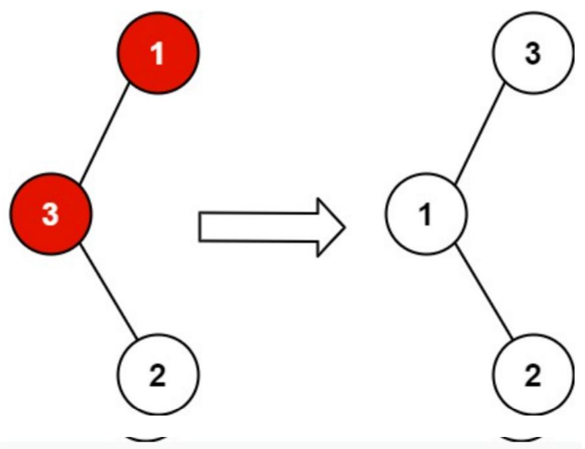
99. Recover Binary Search Tree

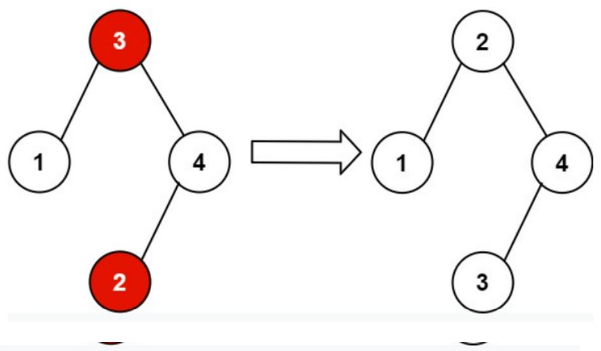
03 April 2022 05:41 PM

You are given the `root` of a binary search tree (BST), where the values of **exactly** two nodes of the tree were swapped by mistake. *Recover the tree without changing its structure.*

Example 1:



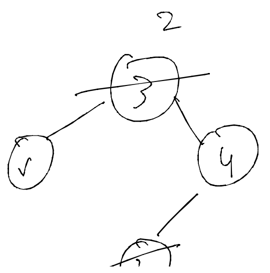
```
Input: root = [1,3,null,null,2]
Output: [3,1,null,null,2]
Explanation: 3 cannot be a left child of 1 because 3 >
1. Swapping 1 and 3 makes the BST valid.
```

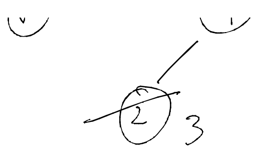


```
Input: root = [3,1,4,null,null,2]
Output: [2,1,4,null,null,3]
Explanation: 2 cannot be in the right subtree of 3
because 2 < 3. Swapping 2 and 3 makes the BST valid.
```

Recover BST

→ Two nodes swapped





Brute force

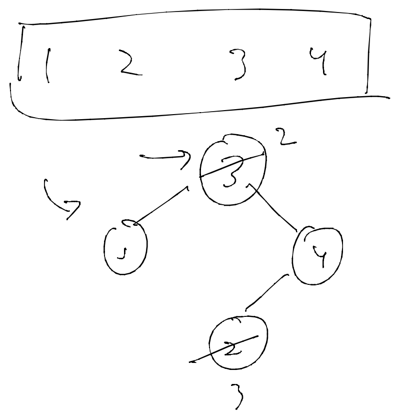
Inorder Traversal

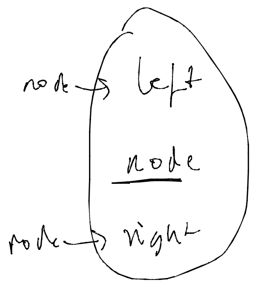
$\downarrow$

Sort

$\downarrow$

Correct Inorder





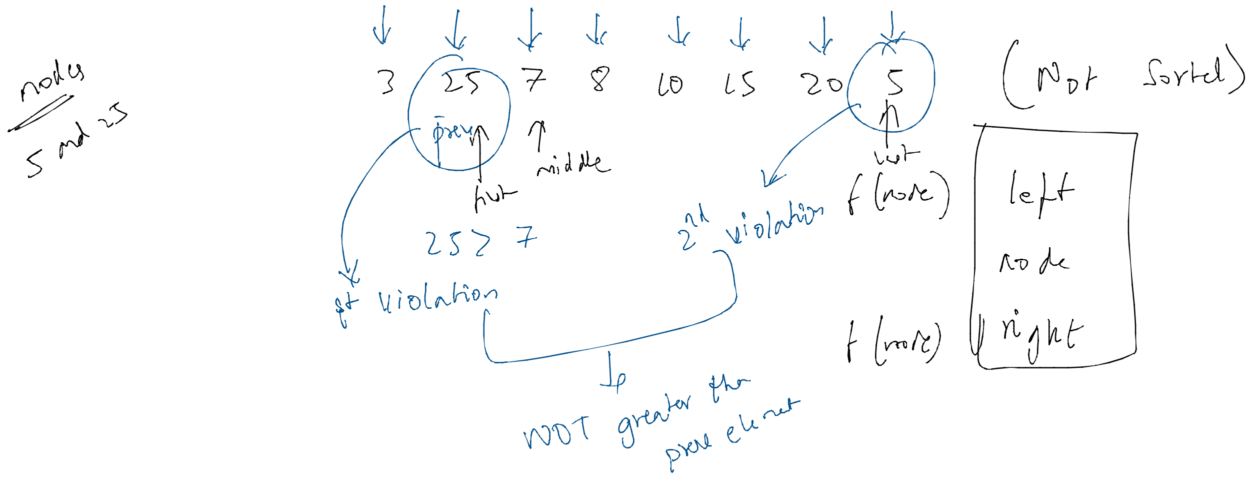
T.C $\Rightarrow O(N) + N\log N$

S.C $\Rightarrow O(N)$

2) Better Solution :

Swap can have 2 cases

1. Swapped nodes are not adjacent

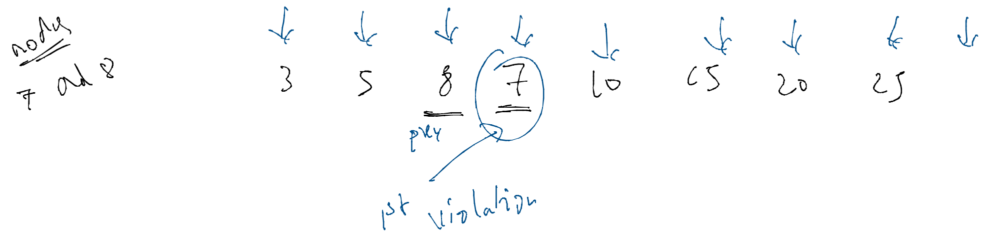


You have stored two violation and then swap it.

You have stored two violation and then swap it.

if there is not second violation ↓ swap(first, last) just storing.

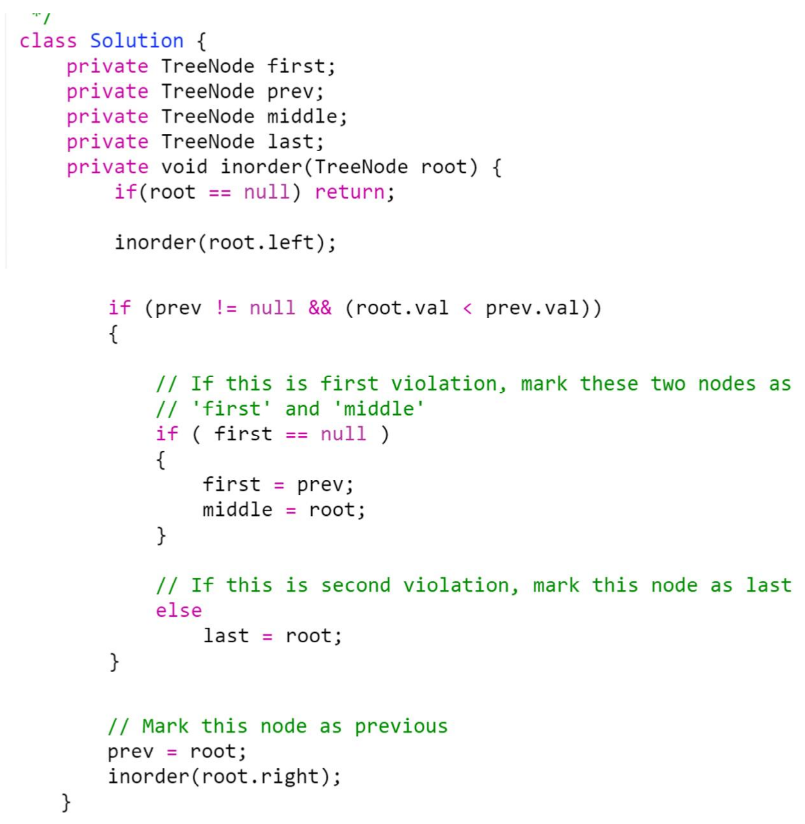
2. Swapped nodes are adjacent



no second violation, so it's a adjacent pair so just swap them.

$T.C \Rightarrow O(N)$

$S.C \Rightarrow O(1)$



```
*/
class Solution {
    private TreeNode first;
    private TreeNode prev;
    private TreeNode middle;
    private TreeNode last;
    private void inorder(TreeNode root) {
        if(root == null) return;

        inorder(root.left);


       if (prev != null && (root.val < prev.val))
       {

           // If this is first violation, mark these two nodes as
           // 'first' and 'middle'
           if ( first == null )
           {
               first = prev;
               middle = root;
           }

           // If this is second violation, mark this node as last
           else
               last = root;
       }

       // Mark this node as previous
       prev = root;
       inorder(root.right);
    }
```

```java
public void recoverTree(TreeNode root) {
    first = middle = last = null;
    prev = new TreeNode(Integer.MIN_VALUE);
    inorder(root);
    if(first!=null && last!=null) {
        int t = first.val;
        first.val = last.val;
        last.val = t;
    }
    else if(first!=null && middle!=null)  {
        int t = first.val;
        first.val = middle.val;
        middle.val = t;
    }
}
```