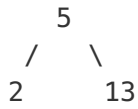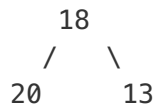## Convert BST to Greater Tree

Given a Binary Search Tree (BST), convert it to a Greater Tree such that every key of the original BST is changed to the original key plus sum of all keys greater than the original key in BST.

Example:

Input: The root of a Binary Search Tree like this:

```
        5
      /   \
     2     13
```

Output: The root of a Greater Tree like this:

```
        18
      /   \
    20     13
```

Solution 1

Since this is a BST, we can do a reverse inorder traversal to traverse the nodes of the tree in descending order. In the process, we keep track of the running sum of all nodes which we have traversed thus far.

```java
public class Solution {

    int sum = 0;

    public TreeNode convertBST(TreeNode root) {
        convert(root);
        return root;
    }

    public void convert(TreeNode cur) {
        if (cur == null) return;
        convert(cur.right);
        cur.val += sum;
        sum = cur.val;
        convert(cur.left);
    }

}
```

written by compton_scatter original link here

Solution 2

Idea: `Reversely` traverse the tree and keep a sum of all previously visited values. Because its a BST, values seen before are all greater than current `node.val`. That's what we want according to the problem.

```java
public class Solution {
    int sum = 0;

    public TreeNode convertBST(TreeNode root) {
        if (root == null) return null;

        convertBST(root.right);

        root.val += sum;
        sum = root.val;

        convertBST(root.left);

        return root;
    }
}
```

written by shawngao original link here

Solution 3

Reversed inorder traversal.

```java
public class Solution {
    public TreeNode convertBST(TreeNode root) {
        if(root == null) return null;
        DFS(root, 0);
        return root;
    }

    public int DFS(TreeNode root, int preSum){
        if(root.right != null) preSum = DFS(root.right, preSum);
        root.val = root.val + preSum;
        return (root.left != null) ? DFS(root.left, root.val) : root.val;
    }
}
```

written by fallcreek original link here