# Convert a BST to a Binary Tree such that sum of all greater keys is added to every key

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

Examples:

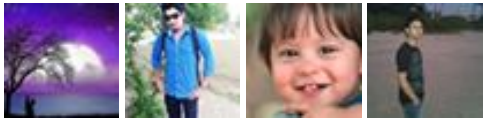
```
Input: Root of following BST
              5
            /   \
           2     13

Output: The given BST is converted to following Binary Tree
              18
            /   \
          20     13
```

Source: Convert a BST

**Solution:** Do reverse Inoorder traversal. Keep track of the sum of nodes visited so far. Let this sum be *sum*. For every node currently being visited, first add the key of this node to *sum*, i.e. *sum = sum + node->key*. Then change the key of current node to *sum*, i.e., *node->key = sum*. When a BST is being traversed in reverse Inorder, for every key currently being visited, all keys that are already visited are all greater keys.

```
// Program to change a BST to Binary Tree such that key of a node beco
// original key plus sum of all greater keys in BST
#include <stdio.h>
```





```
#include <stdlib.h>

/* A BST node has key, left child and right child */
struct node
{
    int key;
    struct node* left;
    struct node* right;
};

/* Helper function that allocates a new node with the given key and
   NULL left and right  pointers.*/
struct node* newNode(int key)
{
    struct node* node = (struct node*)malloc(sizeof(struct node));
    node->key = key;
    node->left = NULL;
    node->right = NULL;
    return (node);
}
```

```
// A recursive function that traverses the given BST in reverse inorde
// for every key, adds all greater keys to it
void addGreaterUtil(struct node *root, int *sum_ptr)
{
    // Base Case
    if (root == NULL)
        return;

    // Recur for right subtree first so that sum of all greater
    // nodes is stored at sum_ptr
    addGreaterUtil(root->right, sum_ptr);

    // Update the value at sum_ptr
    *sum_ptr = *sum_ptr + root->key;

    // Update key of this node
    root->key = *sum_ptr;

    // Recur for left subtree so that the updated sum is added
    // to smaller nodes
    addGreaterUtil(root->left, sum_ptr);
}

// A wrapper over addGreaterUtil().  It initializes sum and calls
// addGreaterUtil() to recursivel upodate and use value of sum
void addGreater(struct node *root)
```





## Popular Posts

```
{
    int sum = 0;
    addGreaterUtil(root, &sum);
}

// A utility function to print inorder traversal of Binary Tree
void printInorder(struct node* node)
{
    if (node == NULL)
        return;
    printInorder(node->left);
    printf("%d ", node->key);
    printInorder(node->right);
}

// Driver program to test above function
int main()
{
    /* Create following BST
              5
           /   \
          2     13  */
    node *root = newNode(5);
    root->left = newNode(2);
    root->right = newNode(13);

    printf(" Inorder traversal of the given tree\n");
    printInorder(root);

    addGreater(root);

    printf("\n Inorder traversal of the modified tree\n");
    printInorder(root);

    return 0;
}
```

Output:

```
 Inorder traversal of the given tree
2 5 13
 Inorder traversal of the modified tree
20 18 13
```

Time Complexity: O(n) where n is the number of nodes in given Binary Search Tree

Time Complexity: O(h) where h is the number of nodes in given Binary Search Tree.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above



## Related Tpoics:

- Print a Binary Tree in Vertical Order | Set 2 (Hashmap based Method)
- Print Right View of a Binary Tree
- Red-Black Tree | Set 3 (Delete)
- Construct a tree from Inorder and Level order traversals
- Print all nodes at distance k from a given node
- Print a Binary Tree in Vertical Order | Set 1
- Interval Tree
- Check if a given Binary Tree is height balanced like a Red-Black Tree



## Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...

Backtracking | Set 7 (Sudoku) · 35 minutes ago

**RVM** Can someone please elaborate this Qs from above...

Flipkart Interview | Set 6 · 55 minutes ago

**Vishal Gupta** I talked about as an Interviewer in general,...

Software Engineering Lab, Samsung Interview | Set 2 · 55 minutes ago

**@meya** Working solution for question 2 of 4f2f round....

Amazon Interview | Set 53 (For SDE-1) · 1 hour ago

sandeep void rearrange(struct node *head) {...

Given a linked list, reverse alternate nodes and append at the end · 2 hours ago

Neha I think that is what it should return as, in...

Find depth of the deepest odd level leaf node · 3 hours ago

**Writing code in comment?** Please use **ideone.com** and share the link here.

## 14 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

**Suryabhan Singh** • 7 months ago

```
void addgreatorsum(struct node *s)
{
    static int pre=0;
    if(!s)
        return ;
    else
    {

        addgreatorsum(s->r);
        s->data+=pre;
        pre=s->data;
        addgreatorsum(s->l);

    }
}
```

Reply • Share ›

**ubiquitous** • 9 months ago

i feel pointer to primitive type here it's int is similar to static variable. So we car
node.value= node.value+right(here right is the sum of the values greater than
atleast return left value Please refer to the code below

atlast return left value Please refer to the code below:

[sourcecode language="java"]
int changeToBinaryTree(TreeNode t, int sum)
{
if(t==null)
{
return sum;
}
int right = changeToBinaryTree(t.right, sum);
t.val = right + t.val;
int left = changeToBinaryTree(t.left, t.val);
return left;
}

1 ▲ | ▼ • Reply • Share ›

**kush** • 11 months ago

```
 tree *sum(tree *root)
{
        static int su=0;
        if(!root)return NULL;
        root->right=sum(root->right);
        su+=root->data;
        root->data=su;
        root->left=sum(root->left);
        return root;
}
```

▲ | ▼ • Reply • Share ›

**abhishek08aug** • 11 months ago

Intelligent :D

1 ^ | v • Reply • Share ›

**Prabodh Panigrahy** • a year ago

No, we will have a separate counter which holds the &#039sum&#039 till the l
with the updated values of the node. Or a simple solution is to add the last visi

^ | v • Reply • Share ›

**sush** • a year ago

Easy implementation

```
 //returns sum of all nodes in tree rooted at "root"
int addGreater(struct node *root)
{
        if(root==NULL)
        return 0;
        root->key+=addGreater(root->right);
        return root->key+addGreater(root->left);
}
```

^ | v • Reply • Share ›

**Gopal** • a year ago

//where root is root of Binary Search Tree
//method call
addAllGreaterKeys(root, 0);

[sourcecode language="JAVA"]
public static int addAllGreaterKeys(Node node,int value)
{
if(node == null)
return 0;

```
if(node.right == null && node.left == null)
{
node.n += value;
return node.n;
}

int rightSum = addAllGreaterKeys(node.right,value);

int leftSum = addAllGreaterKeys(node.left, rightSum + node.n);

node.n += rightSum;

return leftSum;
}
```



**go4gold** • a year ago

```
void addtilllarge(node *p)
{
static int sum;
if(p==NULL)
return;
else
{
addtilllarge(p->right);
sum+=p->data;
p->data=sum;
addtilllarge(p->left);
}
}
```

```
/* Paste your code here (You may delete these lines if not writing c
```

**Shyam Raj** • a year ago

Please correct me folks if am wrong here:

I feel, we first need to traverse the left sub-tree and then the right sub-tree. The
begin to traverse up from the last right sub-tree and keep updating the values;
tree, when we try to add the greater values (which happen to be on the right su
have been already modified.

In simple terms, when we try to add greater values to the left sub-tree, we wou
right sub-tree, since they&#039ve been already modified.

I hope am making some sense here.



**Naren** • a year ago

I think the below code will work..

```
alterNode(Node root, Node Parent,int leftorright)
{
if (root ==null)
return 0;

sum = root->data;
a = alterNode(root->right,root,1);
sum = sum + a;
if (leftorright==0)
sum = sum+parent->data;
root->data = sum;

b = alterNode(root->left,root,0);

if (b!=0)
```

```
if (b!=0)
return b;
else
return sum;
}

alterNode(root,null,1);
```



**Ankush** • a year ago

The above code is not working forme.Here are some modification(Java code).

```
[sourcecode language="JAVA"]
public class Test {
static class Node {
int data;
Node left;
Node right;
}

public static Node newNode(int data) {
Node node = new Node();
node.data = data;
node.left = null;
node.right = null;

return (node);
}
```



**Gopal** ➦ Ankush • a year ago

Correct me if I am wrong, everything looks great, however, in your cod
your left nodes.

• Reply • Share ›

**Kartik** Ankush • a year ago

sum is passed by value in your code. You need to pass it by reference
Java

• Reply • Share ›

**vamshi** • a year ago

```
/* Paste your code here (You may delete these lines if not writing co
This should also generate correct output,If I understand the question

public void alterNode(){
                alterNode( root);
        }
        private int alterNode(BSTNode root){
                if (root == null)
                        return 0;
                int lval=alterNode(root.left);
                int rval=alterNode(root.right);
                int sum=lval+rval+ root.key;
                root.key=root.key+rval;
                return sum;

        }
```

• Reply • Share ›