[Range Sum of BST](https://leetcode.com/problems/range-sum-of-bst/)

**package** unsolvedpackage;

**public** **class** RangeSumBST {

**private** **static** **int** *sum* = 0;

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

TreeNode root = **new** TreeNode(10);

root.left = **new** TreeNode(5);

root.left.left = **new** TreeNode(3);

root.left.right = **new** TreeNode(7);

root.right = **new** TreeNode(15);

root.right.right = **new** TreeNode(18);

System.***out***.println(*rangeSumBST*(root,7,15));

}

**public** **static** **int** rangeSumBST(TreeNode root, **int** L, **int** R) {

**if**(root == **null**)

**return** 0; //checking if tree is empty

*dfs*(root.right, L, R); //depth first search to find the sum of the range

**return** *sum*;

}

**public** **static** **void** dfs(TreeNode node, **int** L, **int** R) {

**if**(node != **null**) {

**if**(L <= node.val && R >= node.val)//check if the current node lies in the range. If so add to the sum

*sum* += node.val;

**if**(L < node.val) //if L is less than node value, then need to check the left subtree

*dfs*(node.left , L , R);

**if**(R > node.val) //if R is greater than node value, then need to check the right subtree

*dfs*(node.right , L , R);

}

}

}

**class** TreeNode {

**int** val;

TreeNode left;

TreeNode right;

TreeNode(**int** x) { val = x; }

}

Time Complexity : O(n) , n is no. of nodes in BST

Space Complexity : O(1)