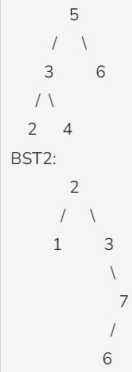
# Merge two BST 's

Difficulty: **Medium** Accuracy: **64.95%** Submissions: **86K+** Points: **4**

Given two BSTs, return elements of merged BSTs in **sorted** form.

**Examples :**

**Input:**
BST1:



BST2:



**Output:** 1 2 2 3 3 4 5 6 6 7
**Explanation:** After merging and sorting the two BST we get 1 2 2 3 3 4 5 6 6 7.

**Input:**
BST1:

BST2:



**Output:** 2 5 6 8 9 10 11 12
**Explanation:** After merging and sorting the two BST we get 2 5 6 8 9 10 11 12.

**Expected Time Complexity:** O((m+n)*log(m+n))
**Expected Auxiliary Space:** O(Height of BST1 + Height of BST2 + m + n)

**Constraints:**
$1 \leq$ Number of Nodes, value of nodes $\leq 10^5$

Try more examples

Seen this question in a real interview before ? Yes No

Topic Tags

Related Articles

Report An Issue
If you are facing any issue on this page. Please let us know.



Java (1.8) Average Time: 30m Start Timer

```java
//{ Driver Code Starts
// Initial Template for Java

/*package whatever //do not write package name here */

import java.io.*;
import java.math.*;
import java.util.*;

class Node {
    int data;
    Node left, right;

    public Node(int d) {
        data = d;
        left = right = null;
    }
}

class GFG {
    static Node buildTree(String str) {
        // Corner Case
        if (str.length() == 0 || str.equals('N')) return null;
        String[] s = str.split(" ");

        Node root = new Node(Integer.parseInt(s[0]));
        Queue<Node> q = new LinkedList<Node>();
        q.add(root);

        // Starting from the second element
        int i = 1;
        while (!q.isEmpty() && i < s.length) {
            // Get and remove the front of the queue
            Node currNode = q.remove();
```

Custom Input Compile & Run Submit