Merge Two Balanced Binary Search Trees

**Method 2 (Merge Inorder Traversals) **

1. Do inorder traversal of first tree and store the traversal in one temp array arr1[]. This step takes O(m) time.
2. Do inorder traversal of second tree and store the traversal in another temp array arr2[]. This step takes O(n) time.
3. The arrays created in step 1 and 2 are sorted arrays. Merge the two sorted arrays into one array of size m + n. This step takes O(m+n) time.
4. Construct a balanced tree from the merged array using the technique discussed in this post. This step takes O(m+n) time.
   Time complexity of this method is O(m+n) which is better than method 1. This method takes O(m+n) time even if the input BSTs are not balanced.
   Following is implementation of this method.

```
class Node:
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None

# A utility unction to merge two sorted arrays into one
# Time Complexity of below function: O(m + n)
# Space Complexity of below function: O(m + n)
def merge_sorted_arr(arr1, arr2):
    arr = []
    i = j = 0
    while i < len(arr1) and j < len(arr2):
        if arr1[i] <= arr2[j]:
            arr.append(arr1[i])
            i += 1
        else:
            arr.append(arr2[j])
            j += 1
    while i < len(arr1):
        arr.append(arr1[i])
        i += 1
    while i < len(arr2):
        arr.append(arr2[j])
        j += 1
```

```
    return arr

# A helper function that stores inorder
# traversal of a tree in arr
def inorder(root, arr = []):
    if root:
        inorder(root.left, arr)
        arr.append(root.val)
        inorder(root.right, arr)

# A utility function to insert the values
# in the individual Tree
def insert(root, val):
    if not root:
        return Node(val)
    if root.val == val:
        return root
    elif root.val > val:
        root.left = insert(root.left, val)
    else:
        root.right = insert(root.right, val)
    return root

# Converts the merged array to a balanced BST
# Explanation of the below code:
# https://www.geeksforgeeks.org/sorted-array-to-balanced-bst/
def arr_to_bst(arr):
    if not arr:
        return None
    mid = len(arr) // 2
    root = Node(arr[mid])
    root.left = arr_to_bst(arr[:mid])
    root.right = arr_to_bst(arr[mid + 1:])
    return root

if __name__=='__main__':
    root1 = root2 = None

    # Inserting values in first tree
    root1 = insert(root1, 100)
    root1 = insert(root1, 50)
    root1 = insert(root1, 300)
    root1 = insert(root1, 20)
```

```
    root1 = insert(root1, 70)

    # Inserting values in second tree
    root2 = insert(root2, 80)
    root2 = insert(root2, 40)
    root2 = insert(root2, 120)
    arr1 = []
    inorder(root1, arr1)
    arr2 = []
    inorder(root2, arr2)
    arr = merge_sorted_arr(arr1, arr2)
    root = arr_to_bst(arr)
    res = []
    inorder(root, res)
    print('Following is Inorder traversal of the merged tree')
    for i in res:
      print(i, end = ' ')
```