```
/*
给你 root1 和 root2 这两棵二叉搜索树。

请你返回一个列表，其中包含 两棵树 中的所有整数并按 升序 排序。.



示例 1：

输入：root1 = [2,1,4], root2 = [1,0,3]
输出：[0,1,1,2,3,4]

示例 2：

输入：root1 = [0,-10,10], root2 = [5,1,7,0,2]
输出：[-10,0,0,1,2,5,7,10]

示例 3：

输入：root1 = [], root2 = [5,1,7,0,2]
输出：[0,1,2,5,7]

示例 4：

输入：root1 = [0,-10,10], root2 = []
输出：[-10,0,10]

示例 5：

输入：root1 = [1,null,8], root2 = [8,1]
输出：[1,1,8,8]

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/all-elements-in-two-binary-search-trees
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析：BST的中序遍历的结果就是升序排列

- 方法一：用两个数组分别存储两个BST的中序访问结果，然后对这两个数组进行归并即可.
    - 缺点：需要额外空间.

方法一：C++_DFS+Merge

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
```

```
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution
{
    private:
        void midOrderTravel(TreeNode* node,vector<int>& vi)
        {
            if(node==NULL)
            {
                return ;
            }
            midOrderTravel(node->left,vi);
            vi.push_back(node->val);
            midOrderTravel(node->right,vi);
        }

    public:
        vector<int> getAllElements(TreeNode* root1, TreeNode* root2)
        {
            vector<int> temp1           ;
            vector<int> temp2           ;
            vector<int> ret_val         ;
            int         length1 =   0   ;
            int         length2 =   0   ;
            int         i       =   0   ;
            int         j       =   0   ;.


            midOrderTravel(root1,temp1);
            midOrderTravel(root2,temp2);
            length1 = temp1.size();
            length2 = temp2.size();

            while(1)
            {
                if(i < length1 && j < length2)
                {
                    if(temp1[i] < temp2[j])
                    {
                        ret_val.push_back(temp1[i]) ;
                        i++;
                    }
                    else
                    {
                        ret_val.push_back(temp2[j]) ;
                        j++;
                    }
                }
                else if(i < length1)
                {
                    ret_val.push_back(temp1[i]) ;
                    i++;
                }
                else if(j < length2)
```

```
                {
                    ret_val.push_back(temp2[j]) ;
                    j++;
                }
                else
                {
                    break;
                }
            }
            return ret_val;
        }
};

/*
执行结果：
通过
显示详情
执行用时 :224 ms, 在所有 C++ 提交中击败了92.91% 的用户
内存消耗 :63.5 MB, 在所有 C++ 提交中击败了6.13%的用户
*/

```

AlimyBreak
2020.01.21(2020年第06周)