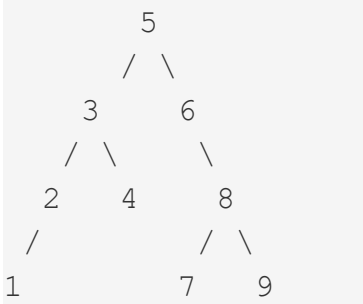
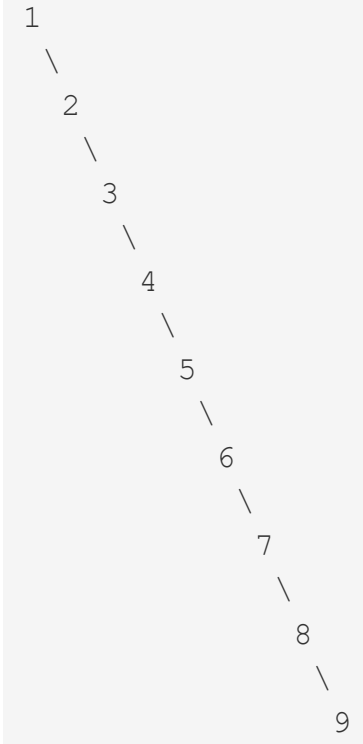
给定一个树，**按中序遍历**重新排列树，使树中最左边的结点现在是树的根，并且每个结点没有左子结点，只有一个右子结点。

示例：

输入： `[5,3,6,2,4,null,8,1,null,null,null,7,9]`



输出： `[1,null,2,null,3,null,4,null,5,null,6,null,7,null,8,null,9]`



提示：

1. 给定树中的结点数介于 1 和 100 之间。
2. 每个结点都有一个从 0 到 1000 范围内的唯一整数值。

思路，直接遍历，得到循序，新增节点，进行排列。自己想多了，想不用数组，直接进行链表操作，结果时间超限。

代码

```
/**
 * 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 {
public:
    void preTrav(TreeNode* cur,vector<int>& inorder){
        if(cur==NULL)return;
        preTrav(cur->left,inorder);
        inorder.push_back(cur->val);
        preTrav(cur->right,inorder);
    }

    TreeNode* increasingBST(TreeNode* root) {
        vector<int> inorder;
        preTrav(root,inorder);
        int n=inorder.size();
        if(n==0)return root;
        root=new TreeNode(inorder[0]);
        TreeNode* cur=root;

        for(int i=1;i<n;i++){
            cur->right=new TreeNode(inorder[i]);
            cur=cur->right;
        }
        return root;
```

```
    }

};
```

自己的代码

```
/**
 * 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 {
public:
    TreeNode* dfs(TreeNode* root,bool flag,TreeNode* &node1){
        if(root==NULL){
            return NULL;
        }
        TreeNode* node=root;
        if(root->left!=NULL){
            node=dfs(root->left,flag,node1);
            flag=false;
            root->left=NULL;
            node->right=root;
            node=node->right;
        }
        if(flag){
            node1=root;
        }
        if(root->right!=NULL){
            root->right=increasingBST(root->right);
            TreeNode* node2;
```

```
            dfs(root->right,true,node2);
            dfs=node2;
        }
        return node;
    }
    TreeNode* increasingBST(TreeNode* root) {
        TreeNode* node=root;
        dfs(root,true,node);
        return node;
    }
};
```