给定二叉搜索树（BST）的根节点和要插入树中的值，将值插入二叉搜索树。 返回插入后二叉搜索树的根节点。 保证原始二叉搜索树中不存在新值。
注意，可能存在多种有效的插入方式，只要树在插入后仍保持为二叉搜索树即可。 你可以返回任意有效的结果。
例如，
给定二叉搜索树：

```
        4
       / \
      2   7
     / \
    1   3
```

和 插入的值：5
你可以返回这个二叉搜索树:

```
         4
       /   \
      2     7
     / \   /
    1   3 5
```

或者这个树也是有效的:

```
         5
       /   \
      2     7
     / \
    1   3
         \
          4
```

思路，因为瞎鸡儿插入都可以，那么按照二叉树的正常思路，进行遍历，到最后，再插入就可以，改题目若是想要升级，就把搜索二叉树改为平衡搜索二叉树，构造平衡搜索的树。

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:
    bool dfs(TreeNode* root, int val){
        if(root->val>val){
            if(root->left!=NULL){
                if(dfs(root->left,val)){
                    return true;
                }
            }else{
                TreeNode* node=new TreeNode(val);
                root->left=node;
            }

        }else{
            if(root->right!=NULL){
                if(dfs(root->right,val)){
                    return true;
                }
            }else{
                TreeNode* node=new TreeNode(val);
                root->right=node;
            }

        }
        return false;
    }
    TreeNode* insertIntoBST(TreeNode* root, int val) {
        if(root==NULL){
            TreeNode* node=new TreeNode(val);
            return node;
        }
        dfs(root,val);
        return root;
    }
};
```