题目描述

输入一棵二叉搜索树，将该二叉搜索树转换成一个排序的双向链表。要求不能创建任何新的结点，只能调整树中结点指针的指向.

代码(我的)：class Solution {

public:

void buildseq(TreeNode\* pRootOfTree)

{

if (pRootOfTree != NULL){ //各元素依次入栈

buildseq(pRootOfTree->left);

node.push\_back(pRootOfTree);

buildseq(pRootOfTree->right);

}

}

TreeNode\* Convert(TreeNode\* pRootOfTree)

{

int i;

buildseq(pRootOfTree);

if (node.size() == 0)head=NULL; //修改前后邻居

else{

head = node[0];

for (i = 0; i < node.size()-1; i++){

node[i]->right = node[i + 1];

node[i + 1]->left = node[i];

}

}

return head;

}

vector<TreeNode \*>node;

TreeNode \* head=NULL;

};

思路分析：利用搜索二叉树的性质依次将排列数的指针入栈，然后再修改前后邻居。

其他代码(1):

|  |
| --- |
| class Solution {  public:      TreeNode\* Convert(TreeNode\* pRootOfTree){      if (pRootOfTree == 0)   return 0; //空树①      TreeNode \*head = 0;      if (pRootOfTree->left == 0 && pRootOfTree->right == 0)return pRootOfTree;      TreeNode\* lhead = 0;  TreeNode\* rhead = 0;      if (pRootOfTree->left != 0) {          lhead = Convert(pRootOfTree->left); //返回左子树的头结点④          head = lhead;          while (lhead->right)lhead = lhead->right;          lhead->right = pRootOfTree; //左子树进行链接，直接向右搜索          pRootOfTree->left = lhead;      }      else head = pRootOfTree; //⑤      if (pRootOfTree->right != 0) { //能到这里说明只有根的左子树为空，则根结点肯是这这颗树的lead          rhead = Convert(pRootOfTree->right);          pRootOfTree->right = rhead;          rhead->left = pRootOfTree;      }      return head; //将子树的最左结点一直返回  同样返回右子树的最左结点，只是返回之后利用方法不一样(此时左子树已经是部分有序的双向链表)  } //直到根结点  };  思路分析：    特殊说明：   1. 树的递归过程中遇到不同的条件就进行返回的顺序往往不能颠倒，例如①-⑥不能颠倒 2. 在递归过程中，左子树的最右结点不需要返回，因为根结点的左子树已经是有序的双向链表，   直接向右搜索就行 |

