题目描述：

输入一个整数数组，判断该数组是不是某二叉搜索树的后序遍历的结果。如果是则输出Yes,否则输出No。假设输入的数组的任意两个数字都互不相同。

代码(我的):

class Solution {

public:

Bool VerifySquenceOfBST(vector<int> sequence) {

int size = sequence.size();

if(size>0)check(0,size-1,sequence);

if (size == 0)flag = 0;

return flag;

}

Void check(int start, int end, vector<int>seq) {

int i, j,mark;

If (start! = end){ //只要有子树就往下递归验证

if (seq[end] < seq[start]){//没有左子树

for (i = start; i <= end - 1; i++){

if (seq[i] < seq[end]){

flag = 0;

break;

}

}

If (flag==1) check(start, end - 1, seq);

}

else if (seq[end - 1] < seq[end]){//没有右子树

for (i = start; i <= end - 1; i++){

if (seq[i]>seq[end]){

flag = 0;

break;

}

}

if(flag==1)check(start, end - 1, seq);

}

else {//左右子树都在

for (i = start; i <= end - 1; i++){ //切割左右子树

if (seq[i] > seq[end]){ mark = i; break; }

}

for (i = mark; i <= end - 1; i++){

if (seq[i] < seq[end]){

flag = 0;

break;

}

}

if (flag == 1){

check(start, mark - 1, seq);//验证左子树

check(mark, end-1, seq);//验证右子树

}

}

}

}//end for check

int flag = 1;

};

分析:分析每种可能出现的情况:A、1.左子树为空，2.右子树为空，3.左右子树均不为空的情况

B、然后在详细考虑每种情况下不是二叉搜索树的后序序列的情况

C、一定要做好各种边缘情况的处理，比如说该序列为空的时候 if (size == 0)flag = 0;

代码2（其它）：

class Solution {

public:

bool VerifySquenceOfBST(vector<int> sequence) {

int i=0,j=0;

j=sequence.size();

if(j==0) return false;

while(--j){

while(sequence[i]<sequence[j]) i++;

while(sequence[i]>sequence[j]) i++;

if(i<j) return false;

i=0;

}

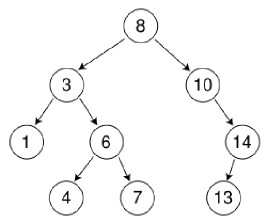
return true;

}

};

分析:

搜索二叉树的结构:



该算法每次以根结点为比较对象，每次也只有根结点的右孩子的值才比自己大，又因为是后续遍历，所以根结点都会是序列的最后一个，每次—j的结点值会越来越大。依次比较如果碰到第一个比自己大的结点的话算法就认为这是根结点的右子树的开始，一旦后续有结点比自己小的话就一定不是满足条件的序列。