验证序列是否为二叉搜索树BST的前序或后序遍历

# 验证序列是否为二叉搜索树BST的后序遍历

## 题目描述

/\*\*\*\*剑指Offer33：验证数组是否是某二叉搜索树的前序遍历或后序遍历序列\*\*\*/

\* 题目介绍：

\* 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。

## 思路分析

\* 思路分析：根据二叉搜索树的特点，根节点的左子树的值都小于等于根节点；

\* 根节点的右子树的值都大于根节点。

\* 一个二叉搜索树的后序遍历序列的最后一个元素为根节点的值，那么根节点之前

\* 的可以分成两部分，第一部分都小于根节点的值，第二部分都大于根节点的值。

\* 然后对第一部分和第二部分再进行递归调用即可。

\*/

## Java代码

public boolean **VerifySquenceOfBSTPostOrder**(int[] sequence) {

if(sequence==null||sequence.length==0) return false;

return VerifySquenceOfBSTPostOrder(sequence,0,sequence.length-1);

}

/\*\*

\* 递归调用方法

\*/

private boolean VerifySquenceOfBSTPostOrder(int[] sequence,int start,int end){

if(end - start < 2) return true;//1个或2个元素一定满足二叉搜索树；3个就不一定了

int rootValue = sequence[end];

int leftIndex = start;

while(leftIndex < end && sequence[leftIndex]<= rootValue) leftIndex++;

int rightIndex = leftIndex;

while(sequence[rightIndex] > rootValue) rightIndex++;//由于最后一个元素肯定等于rootValue，所以不担心数组索引越界

if(rightIndex != end) return false;

**boolean left = VerifySquenceOfBSTPostOrder(sequence,start,leftIndex-1);**

**boolean right = VerifySquenceOfBSTPostOrder(sequence,leftIndex,end-1);**

return left&&right;

}

# 验证序列是否为二叉搜索树BST的前序遍历

## 题目描述

/\*\*\*\*举一反三：验证数组是否是某二叉搜索树的前序遍历\*\*\*\*\*\*\*\*\*\*/

## 思路分析

\* 分析：前序遍历和后序遍历的区别就是：后序遍历是最后一个元素为根节点，

\* 而前序遍历的第一个元素为根节点，其他的都是相同的。

**防止数组索引越界，很重要。**

## Java代码

public boolean **VerifySquenceOfBSTPreOrder**(int[] sequence) {

if(sequence==null||sequence.length==0) return false;

return VerifySquenceOfBSTPreOrder(sequence,0,sequence.length-1);

}

/\*\*

\* 递归调用方法

\*/

private boolean **VerifySquenceOfBSTPreOrder**(int[] sequence,int start,int end){

if(end - start < 2) return true;//1个或2个元素一定满足二叉搜索树；3个就不一定了

int rootValue = sequence[start];

int leftIndex = start+1;

while(**leftIndex <= end && sequence[leftIndex]<= rootValue**) leftIndex++;

int rightIndex = leftIndex;

while(**rightIndex <= end&&sequence[rightIndex] > rootValue**) rightIndex++;//由于rootValue在头部，尾部一定要控制数组越界问题

if(rightIndex != end+1) return false;

**boolean left = VerifySquenceOfBSTPostOrder(sequence,start+1,leftIndex-1);**

**boolean right = VerifySquenceOfBSTPostOrder(sequence,leftIndex,end);**

**return left&&right;**

}