# 题目

给定一个整数数组，你需要验证它是否是一个二叉搜索树正确的先序遍历序列。

你可以假定该序列中的数都是不相同的。

参考以下这颗二叉搜索树：

5

/ \

2 6

/ \

1 3

示例 1：

输入: [5,2,6,1,3]

输出: false

示例 2：

输入: [5,2,1,3,6]

输出: true

进阶挑战：

您能否使用恒定的空间复杂度来完成此题？

# 分析

要解决这个问题，我们需要验证一个给定的整数数组是否是一个二叉搜索树（BST）的正确先序遍历序列。二叉搜索树的特性是：对于任意节点，其左子树的所有节点值都小于该节点值，右子树的所有节点值都大于该节点值。先序遍历的顺序是“根->左->右”。

解题思路

利用二叉搜索树和先序遍历的特性，我们可以采用单调栈的方法来解决这个问题，具体思路如下：

1、单调递减栈：由于先序遍历中左子树的节点值是递减的，我们可以使用一个栈来模拟左子树的遍历过程，保持栈的单调递减特性。

2、更新下界：当遇到一个大于栈顶的元素时，说明我们开始遍历右子树。此时需要弹出栈中所有小于当前元素的节点，这些节点构成了当前元素的左祖先链，其中最后弹出的节点是当前元素的直接父节点，我们将其作为右子树节点的下界（所有右子树节点必须大于该值）。

3、检查合法性：如果当前元素小于下界，则说明该序列不符合二叉搜索树的先序遍历规则，返回false。

此外，为了优化空间复杂度，我们可以复用原数组来模拟栈的行为，从而实现常数空间复杂度。

代码：

class Solution {

public:

bool verifyPreorder(vector<int>& preorder) {

int low = INT\_MIN; // 记录当前右子树的下界

int i = -1; // 模拟栈顶指针，初始栈为空

for (int num : preorder) {

// 如果当前数字小于下界，说明不满足BST特性

if (num < low) {

return false;

}

// 当遇到大于栈顶的元素时，弹出栈中所有小于当前元素的节点，并更新下界

while (i >= 0 && num > preorder[i]) {

low = preorder[i--];

}

// 将当前元素入栈（复用原数组）

preorder[++i] = num;

}

return true;

}

};

解释：

- 单调栈模拟：通过复用原数组的方式，我们用变量i来模拟栈顶指针。当遍历元素时，若元素小于栈顶（即preorder[i]），则将其入栈（preorder[++i] = num），保持栈的递减特性。

- 下界更新：当遇到大于栈顶的元素时，说明进入了右子树。此时需要弹出所有小于当前元素的栈顶元素，这些元素是当前元素的左祖先，最后弹出的元素成为右子树的下界（low），确保后续元素都大于该下界。

- 合法性检查：如果任何元素小于当前下界，说明该元素不应该出现在右子树中，因此序列不合法，返回false。

这种方法的时间复杂度为O(n)（每个元素入栈和出栈各一次），空间复杂度为O(1)（复用原数组，未使用额外空间），满足进阶挑战的要求。