```swift
import Foundation
/// **剑指 Offer 33. 二叉搜索树的后序遍历序列**
/// 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结果。
/// 如果是则返回 true，否则返回 false。假设输入的数组的任意两个数字都互不相同。
/// **后序遍历定义： [ 左子树 | 右子树 | 根节点 ] ，即遍历顺序为 “左、右、根” 。**
///
/// **二叉搜索树定义：**
/// 左子树中所有节点的值 小于 根节点的值；
/// 右子树中所有节点的值 大于 根节点的值；
/// 其左、右子树也分别为二叉搜索树。
class Solution {
    /// 存储后序遍历的数组用于 recur 函数使用
    private var postorder: [Int] = []
    func verifyPostorder(_ postorder: [Int]) -> Bool {
        self.postorder = postorder
        return recur(0, self.postorder.count-1)
    }
    /// 用于判断是否为二叉搜索树
    /// - Parameters:
    ///   - i: 根节点左子树的起始索引
    ///   - j: 根节点右子树的终止索引
    /// - Returns: 是否为二叉搜索树
    private func recur(_ i: Int, _ j: Int) -> Bool {
        if i >= j { return true }
        /// **关键：对于后序遍历，最后一个值必然为树的总根节点，在此处索引为 j **
        /// 寻找第一个大于根节点的树的索引 m 则可以将数组划分为三个部分
        /// [左子树 (比根节点小), 右子树(比根节点大), 根节点] = [( i, m-1 ), ( m, j-1 ), j]
        var m = i
        while self.postorder[m] < self.postorder[j] { m += 1 }
        /// 判断右子树中所有的节点是否大于根节点
        /// 创建一个临时变量用于遍历右节点，显然如果在遍历过程中右子树小于根节点必然不是搜索树
        /// 同时遍历完后应当满足 p==j
        var p = m
        while self.postorder[p] > self.postorder[j] { p += 1 }
        /// 递归遍历分出来的左右子树判断是否均满足标准的二叉搜索树
        return (p == j) && recur(i, m-1) && recur(m, j-1)
    }
}

let solution = Solution()
print(solution.verifyPostorder([1,6,3,2,5]))
print(solution.verifyPostorder([1,3,2,6,5]))
print(solution.verifyPostorder([1,4,3,2,6,9,8,7,5]))
```