于 1，如果大于 1，说明已经发现整棵树不是平衡二叉树，如果不大于 1，则返回 lH 和 rH 较大的一个。

判断的全部过程请参看如下代码中的 isBalance 方法。在递归函数 getHeight 中，一旦发现不符合平衡二叉树的性质，递归过程会迅速退出，此时返回什么根本不重要。boolean[] res 长度为 1，其功能相当于一个全局的 boolean 变量。

```
public boolean isBalance(Node head) {
        boolean[] res = new boolean[1];
        res[0] = true;
        getHeight(head, 1, res);
        return res[0];
}

public int getHeight(Node head, int level, boolean[] res) {
        if (head == null) {
                return level;
        }
        int lH = getHeight(head.left, level + 1, res);
        if (!res[0]) {
                return level;
        }
        int rH = getHeight(head.right, level + 1, res);
        if (!res[0]) {
                return level;
        }
        if (Math.abs(lH - rH) > 1) {
                res[0] = false;
        }
        return Math.max(lH, rH);
}
```

整个后序遍历的过程中，每个节点最多遍历一次，如果中途发现不满足平衡二叉树的性质，整个过程会迅速退出，没遍历到的节点也不用遍历了，所以时间复杂度为 $O(N)$。

根据后序数组重建搜索二叉树

【题目】

给定一个整型数组 arr，已知其中没有重复值，判断 arr 是否可能是节点值类型为整型的搜索二叉树后序遍历的结果。

进阶：如果整型数组 arr 中没有重复值，且已知是一棵搜索二叉树的后序遍历结果，通过数组 arr 重构二叉树。

【难度】

士 ★☆☆☆

【解答】

原问题的解法。二叉树的后序遍历为先左、再右、最后根的顺序，所以，如果一个数组是二叉树后序遍历的结果，那么头节点的值一定会是数组的最后一个元素。搜索二叉树的性质，所以比后序数组最后一个元素值小的数组会在数组的左边，比数组最后一个元素值大的数组会在数组的右边。比如 arr=[2,1,3,6,5,7,4]，比 4 小的部分为[2,1,3]，比 4 大的部分为[6,5,7]。如果不满足这种情况，说明这个数组一定不可能是搜索二叉树后序遍历的结果。接下来数组划分成左边数组和右边数组，相当于二叉树分出了左子树和右子树，只要递归地进行如上判断即可。

具体过程请参看如下代码中的 isPostArray 方法。

```
public boolean isPostArray(int[] arr) {
        if (arr == null || arr.length == 0) {
                return false;
        }
        return isPost(arr, 0, arr.length - 1);
}

public boolean isPost(int[] arr, int start, int end) {
        if (start == end) {
                return true;
        }
        int less = -1;
        int more = end;
        for (int i = start; i < end; i++) {
                if (arr[end] > arr[i]) {
                        less = i;
                } else {
                        more = more == end ? i : more;
                }
        }
        if (less == -1 || more == end) {
                return isPost(arr, start, end - 1);
        }
        if (less != more - 1) {
                return false;
        }
        return isPost(arr, start, less) && isPost(arr, more, end - 1);
}
```

当前节点没有左子树或没有右子树

不能分割成两个子数组

进阶问题的分析与原问题同理，一棵树的后序数组中最后一个值为二叉树头节点的值，

数组左部分都比头节点的值小，用来生成头节点的左子树，剩下的部分用来生成右子树。

具体过程请参看如下代码中的 posArrayToBST 方法。

```
public class Node {
        public int value;
        public Node left;
        public Node right;

        public Node(int value) {
                this.value = value;
        }
}

public Node posArrayToBST(int[] posArr) {
        if (posArr == null) {
                return null;
        }
        return posToBST(posArr, 0, posArr.length - 1);
}

public Node posToBST(int[] posArr, int start, int end) {
        if (start > end) {
                return null;
        }
        Node head = new Node(posArr[end]);
        int less = -1;
        int more = end;
        for (int i = start; i < end; i++) {
                if (posArr[end] > posArr[i]) {
                        less = i;
                } else {
                        more = more == end ? i : more;
                }
        }
        head.left = posToBST(posArr, start, less);
        head.right = posToBST(posArr, more, end - 1);
        return head;
}
```

判断一棵二叉树是否为搜索二叉树和完全二叉树

【题目】

给定一个二叉树的头节点 head，已知其中没有重复值的节点，实现两个函数分别判断这棵二叉树是否是搜索二叉树和完全二叉树。