在二叉树中找到累加和为指定值的最长路径长度

【题目】

给定一棵二叉树的头节点 head 和一个 32 位整数 sum，二叉树节点值类型为整型，求累加和为 sum 的最长路径长度。路径是指从某个节点往下，每次最多选择一个孩子节点或者不选所形成的节点链。

例如，二叉树如图 3-15 所示。

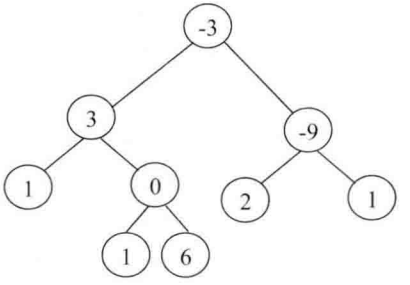

图 3-15

如果 sum=6，那么累加和为 6 的最长路径为：-3，3，0，6，所以返回 4。

如果 sum=-9，那么累加和为-9 的最长路径为：-9，所以返回 1。

注：本题不用考虑节点值相加可能溢出的情况。

【难度】

尉 ★★☆☆

【解答】

在阅读本题的解答之前，请读者先阅读本书“求未排序数组中累加和为规定值的最长子数组长度”问题。针对二叉树，本文的解法改写了这个问题的实现。如果二叉树的节点数为 N，本文的解法可以做到时间复杂度为 $O(N)$，额外空间复杂度为 $O(h)$，其中，h 为二叉树的高度。

具体过程如下：

1．二叉树头节点 head 和规定值 sum 已知；生成变量 maxLen，负责记录累加和等于

sum 的最长路径长度。

2．生成哈希表 sumMap。在“求未排序数组中累加和为规定值的最长子数组长度”问题中也使用了哈希表，功能是记录数组从左到右的累加和出现情况，在遍历数组的过程中，再利用这个哈希表来求得累加和为规定值的最长子数组。sumMap 也一样，它负责记录从 head 开始的一条路径上的累加和出现情况，累加和也是从 head 节点的值开始累加的。sumMap 的 key 值代表某个累加和，value 值代表这个累加和在路径中最早出现的层数。如果在遍历到 cur 节点的时候，我们能够知道从 head 到 cur 节点这条路径上的累加和出现情况，那么求以 cur 节点结尾的累加和为指定值的最长路径长度就非常容易。究竟如何去更新 sumMap，才能够做到在遍历到任何一个节点的时候都能有从 head 到这个节点的路径上的累加和出现情况呢？步骤 3 详细地说明了更新过程。

3．首先在 sumMap 中加入一个记录(0,0)，它表示累加和 0 不用包括任何节点就可以得到。然后按照二叉树先序遍历的方式遍历节点，遍历到的当前节点记为 cur，从 head 到 cur 父节点的累加和记为 preSum，cur 所在的层数记为 level。将 cur.value+preSum 的值记为 curSum，就是从 head 到 cur 的累加和。如果 sumMap 中已经包含了 curSum 的记录，说明 curSum 在上层中已经出现过，那么就不更新 sumMap；如果 sumMap 不包含 curSum 的记录，说明 curSum 是第一次出现，就把(curSum,level)这个记录放入 sumMap。接下来是求解在必须以 cur 结尾的情况下，累加和为规定值的最长路径长度，详细过程这里不再详述，请读者阅读“求未排序数组中累加和为规定值的最长子数组长度”问题。然后是遍历 cur 左子树和右子树的过程，依然按照步骤 3 描述的使用和更新 sumMap。以 cur 为头节点的子树处理完，当然要返回到 cur 父节点，在返回前还有一项重要的工作要做，在 sumMap 中查询 curSum 这个累加和（key）出现的层数（value），如果 value 等于 level，说明 curSum 这个累加和的记录是在遍历到 cur 时加上去的，那就把这一条记录删除；如果 value 不等于 level，则不做任何调整。

4．步骤 3 会遍历二叉树所有的节点，也会求解以每个节点结尾的情况下，累加和为规定值的最长路径长度。用 maxLen 记录其中的最大值即可。

全部求解过程请参看如下代码中的 getMaxLength 方法。

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

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

```
public int getMaxLength(Node head, int sum) {
        HashMap<Integer, Integer> sumMap = new HashMap<Integer, Integer>();
        sumMap.put(0, 0); // 重要
        return preOrder(head, sum, 0, 1, 0, sumMap);
}

public int preOrder(Node head, int sum, int preSum, int level,
                int maxLen, HashMap<Integer, Integer> sumMap) {
        if (head == null) {
                return maxLen;
        }
        int curSum = preSum + head.value;
        if (!sumMap.containsKey(curSum)) {
                sumMap.put(curSum, level);
        }
        if (sumMap.containsKey(curSum - sum)) {
                maxLen = Math.max(level - sumMap.get(curSum - sum), maxLen);
        }
        maxLen = preOrder(head.left, sum, curSum, level + 1, maxLen, sumMap);
        maxLen = preOrder(head.right, sum, curSum, level + 1, maxLen, sumMap);
        if (level == sumMap.get(curSum)) {
                sumMap.remove(curSum);
        }
        return maxLen;
}
```

找到二叉树中的最大搜索二叉子树

【题目】

给定一棵二叉树的头节点 head，已知其中所有节点的值都不一样，找到含有节点最多的搜索二叉子树，并返回这棵子树的头节点。

例如，二叉树如图 3-16 所示。

这棵树中的最大搜索二叉子树如图 3-17 所示。

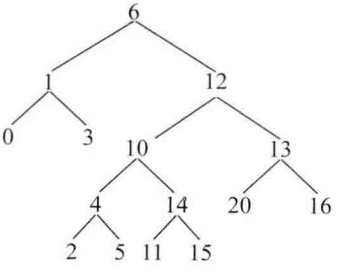

图 3-16

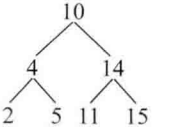

图 3-17