在二叉树中找到一个节点的后继节点

【题目】

现在有一种新的二叉树节点类型如下：

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

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

该结构比普通二叉树节点结构多了一个指向父节点的 parent 指针。假设有一棵 Node 类型的节点组成的二叉树，树中每个节点的 parent 指针都正确地指向自己的父节点，头节点的 parent 指向 null。只给一个在二叉树中的某个节点 node，请实现返回 node 的后继节点的函数。在二叉树的中序遍历的序列中，node 的下一个节点叫作 node 的后继节点。

例如，图 3-40 所示的二叉树。

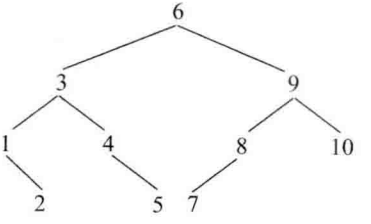

图 3-40

中序遍历的结果为：1，2，3，4，5，6，7，8，9，10

所以节点 1 的后继为节点 2，节点 2 的后继为节点 3，……，节点 10 的后继为 null。

【难度】

尉 ★★☆☆

【解答】

先简单介绍一种时间复杂度和空间复杂度较高但易于理解的方法。既然新类型的二叉树节点有指向父节点的指针，那么一直往上移动，自然可以找到头节点。找到头节点之后，再进行二叉树的中序遍历，生成中序遍历序列，然后在这个序列中找到 node 节点的下一个节点返回即可。如果二叉树的节点数为 N，这种方法要把二叉树的所有节点至少遍历一遍，生成中序遍历的序列还需要大小为 N 的空间，所以该方法的时间复杂度与额外空间复杂度都为 $O(N)$。本书不再详述。

最优解法不必遍历所有的节点，如果 node 节点和 node 后继节点之间的实际距离为 L，最优解法只用走过 L 个节点，时间复杂度为 $O(L)$，额外空间复杂度为 $O(1)$。接下来详细说明最优解法是如何找到 node 的后继节点的。

情况 1：如果 node 有右子树，那么后继节点就是右子树上最左边的节点。

例如，题目所示的二叉树中，当 node 为节点 1、3、4、6 或 9 时，就是这种情况。

情况 2：如果 node 没有右子树，那么先看 node 是不是 node 父节点的左孩子，如果是左孩子，那么此时 node 的父节点就是 node 的后继节点；如果是右孩子，就向上寻找 node 的后继节点，假设向上移动到的节点记为 s，s 的父节点记为 p，如果发现 s 是 p 的左孩子，那么节点 p 就是 node 节点的后继节点，否则就一直向上移动。

例如，题目所示的二叉树中，当 node 为节点 7 时，节点 7 的父节点是节点 8，同时节点 7 是节点 8 的左孩子，此时节点 8 就是节点 7 的后继节点。

再如，题目所示的二叉树中，当 node 为节点 5 时，节点 5 的父节点是节点 4，但是节点 5 是节点 4 的右孩子，所以向上寻找 node 的后继节点。当向上移动到节点 4，节点 4 的父节点是节点 3，但是节点 4 还是节点 3 的右孩子，继续向上移动。当向上移动到节点 3 时，节点 3 的父节点是节点 6，此时终于发现节点 3 是节点 6 的左孩子，移动停止，节点 6 就是 node(节点 5)的后继节点。

情况 3：如果在情况 2 中一直向上寻找，都移动到空节点时还是没有发现 node 的后继节点，说明 node 根本不存在后继节点。

比如，题目所示的二叉树中，当 node 为节点 10 时，一直向上移动到节点 6，此时发现节点 6 的父节点已经为空，说明 node 没有后继节点。

情况 1 和情况 2 遍历的节点就是 node 到 node 后继节点这条路径上的节点；情况 3 遍历的节点数也不会超过二叉树的高度。

最优解的具体过程请参看如下代码中的 getNextNode 方法。

```
public Node getNextNode(Node node) {
        if (node == null) {
                return node;
        }
        if (node.right != null) {
                return getLeftMost(node.right);
        } else {
                Node parent = node.parent;
                while (parent != null && parent.left != node) {
                        node = parent;
                        parent = node.parent;
                }
                return parent;
        }
}

public Node getLeftMost(Node node) {
        if (node == null) {
                return node;
        }
        while (node.left != null) {
                node = node.left;
        }
        return node;
}
```

node有右孩子，则其右子树的最小节点为其后继

若node不是它的父节点的左孩子，则需要一直往上走

为左孩子，则父节点为后继

获取最左节点

在二叉树中找到两个节点的最近公共祖先

【题目】

给定一棵二叉树的头节点 head，以及这棵树中的两个节点 o1 和 o2，请返回 o1 和 o2 的最近公共祖先节点。

例如，图 3-41 所示的二叉树。

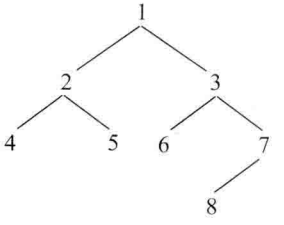

图 3-41

节点 4 和节点 5 的最近公共祖先节点为节点 2，节点 5 和节点 2 的最近公共祖先节点