236. Lowest Common Ancestor of a Binary Tree

link

第一种思路, inOrder缓存所有的节点到map然后判断是不是在里面 time space都是 O(n)

```
int count;
public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
    if(root.val == p.val) return p;
    if(root.val == q.val) return q;
    Map<Integer, Integer> map = new HashMap();
    count = 0;
    helper(root.left, p, q, map);
    int indexP = map.get(p.val), indexQ = map.get(q.val);
    while(((indexP - map.get(root.val)) * (map.get(root.val) - indexQ)) < 0){
        if(indexP > map.get(root.val))
            root = root.right;
        else root = root.left;
    }
    return root;

}
public void helper(TreeNode root, TreeNode p, TreeNode q, Map<Integer, Integer>
map) {
    if(pFound > 0 && qFound > 0){
        if(root == null) return;
        helper(root.left, p, q, map);
        map.put(root.val, ++count);
        helper(root.right, p, q, map);
    }
}
```

第二种思路只往左边走. 判断是不是都在左边, 我最开始的写法 O(n^2) 递归每一次都重新判断了, 递归掌握的不好.

```
int pLeft;
int qLeft;
public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
    while(root != null){
        if(root.val == p.val) return p;
        if(root.val == q.val) return q;
        pLeft = -1;
        qLeft = -1;
        findNode(root.left, p, q);
```

```
            System.out.print(pLeft+""+ qLeft);
            if(pLeft * qLeft < 0) return root;
            if(pLeft == 1 && qLeft == 1)
                root = root.left;
            else
                root = root.right;
        }
        return null;
    }
    public void findNode(TreeNode root, TreeNode p, TreeNode q) {
        if(root == null) return;
        if(root.val == p.val) pLeft = 1;
        if(root.val == q.val) qLeft = 1;
        if(pLeft == 1 && qLeft == 1) return;
        findNode(root.left, p , q);
        findNode(root.right, p, q);
    }
```

考虑加一个return node来决定在左边还是右边. 递归空间复杂度一般是O(h).

```
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
        if(root == null || root == p || root ==q) return root;
        TreeNode left = lowestCommonAncestor(root.left, p, q);
        TreeNode right = lowestCommonAncestor(root.right, p, q);
        if(left == null && right == null)
            return null;
        if(left != null && right != null)
            return root;
        return left == null ? right : left;
    }
```