236 二叉树的最近公共祖先

Label：给定一个二叉树，找到该树中两个指定节点的最近公共祖先。

百度百科中最近公共祖先的定义为：“对于有根树 T 的两个节点 p、q，最近公共祖先表示为一个节点 x，满足 x 是 p、q 的祖先且 x 的深度尽可能大（一个节点也可以是它自己的祖先）。”

p != q
p 和 q 均存在于给定的二叉树中。
所有 Node.val 互不相同

- 存储父节点

```
class Solution {
    Map<Integer, TreeNode> parent = new HashMap<Integer, TreeNode>();
    Set<Integer> visited = new HashSet<Integer>();

    public void dfs(TreeNode root) {
        if (root.left != null) {
            parent.put(root.left.val, root);
            dfs(root.left);
        }
        if (root.right != null) {
            parent.put(root.right.val, root);
            dfs(root.right);
        }
    }

    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q)
{
        dfs(root);
        while (p != null) {
            visited.add(p.val);  // 存储p所有的父节点
            p = parent.get(p.val);// 找到p的父节点
        }

        while (q != null) {  // 遍历q的父节点，找到第一个与p的父节点重合的父节点
            if (visited.contains(q.val)) {
                return q;
            }
            q = parent.get(q.val);
        }
        return null;
    }
}
```

- 递归

```
class Solution {
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q)
{
        if (root == null || root == p || root == q) {  // 是否找到目标节点
            return root;
        }

        TreeNode left = lowestCommonAncestor(root.left, p, q);
        TreeNode right = lowestCommonAncestor(root.right, p, q);

        // 左子树和右子树都没找到，说明两个节点都不在这棵树上
        if (left == null && right == null) return null;

        // 左子树找到一个，右子树找到一个，那么找到确定的公共祖先
        else if (left != null && right != null) return root;

        // 如果一个子树找到一个节点，另一个子树没找到，那必然还没找到的节点出现包含关系，在同一
棵子树上
        else return left == null ? right : left;  // 存在包含关系
    }
}
```