0235. 二叉搜索树的最近公共祖先

ITCharge 大约 2 分钟

- 标签：树、深度优先搜索、二叉搜索树、二叉树
- 难度：中等

题目链接

- 0235. 二叉搜索树的最近公共祖先 - 力扣

题目大意

描述：给定一个二叉搜索树的根节点 `root` ，以及两个指定节点 `p` 和 `q` 。

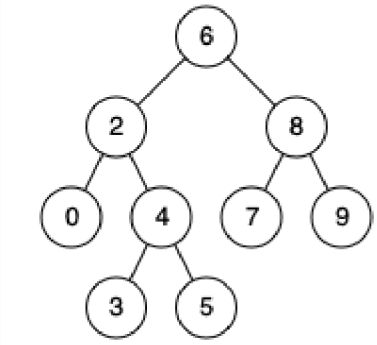
要求：找到该树中两个指定节点的最近公共祖先。

说明：

- **祖先**：若节点 `p` 在节点 `node` 的 `` 树或右子树中，或者 `p == node` ，则称 `node` 是 `p` 的祖先。
- **最近公共祖先**：对于树的两个节点 `p` 、 `q` ，最近公共祖先表示为一个节点 `lca_node` ，满足 `lca_node` 是 `p` 、 `q` 的祖先且 `lca_node` 的深度尽可能大（一个节点也可以是自己的祖先）。
- 所有节点的值都是唯一的。
- `p` 、 `q` 为不同节点且均存在于给定的二叉搜索树中。

示例：

- 示例 1：



img

```py
输入：root = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 8
输出：6
解释：节点 2 和节点 8 的最近公共祖先是 6。
```

- 示例 2：

```py
输入：root = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 4
输出：2
解释：节点 2 和节点 4 的最近公共祖先是 2，因为根据定义最近公共祖先节点可以为节点本身。
```

解题思路

思路 1：递归遍历

对于节点 `p` 、节点 `q` ，最近公共祖先就是从根节点分别到它们路径上的分岔点，也是路径中最后一个相同的节点，现在我们的　　就是求这个分岔点。

我们可以使用递归遍历查找二叉搜索树的最近公共祖先，具体方法如下。

1. 从根节点 `root` 开始遍历。
2. 如果当前节点的值大于 `p` 、 `q` 的值，说明 `p` 和 `q` 应该在当前节点的左子树，因此将当前节点移动到它的左子节点，继续遍历；
3. 如果当前节点的值小于 `p` 、 `q` 的值，说明 `p` 和 `q` 应该在当前节点的右子树，因此将当前节点移动到它的右子节点，继续遍历；
4. 如果当前节点不满足上面两种情况，则说明 `p` 和 `q` 分别在当前节点的左右子树上，则当前节点就是分岔点，直接返回该节点即可。

思路 1：代码

```py
class Solution:
    def lowestCommonAncestor(self, root: 'TreeNode', p: 'TreeNode', q: 'TreeNode') -> 'TreeNode':
        ancestor = root
        while True:
```

```
            if ancestor.val > p.val and ancestor.val > q.val:
                ancestor = ancestor.left
            elif ancestor.val < p.val and ancestor.val < q.val:
                ancestor = ancestor.right
            else:
                break
        return ancestor
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$。其中 n 是二叉搜索树的节点个数。
- **空间复杂度**：$O(1)$。