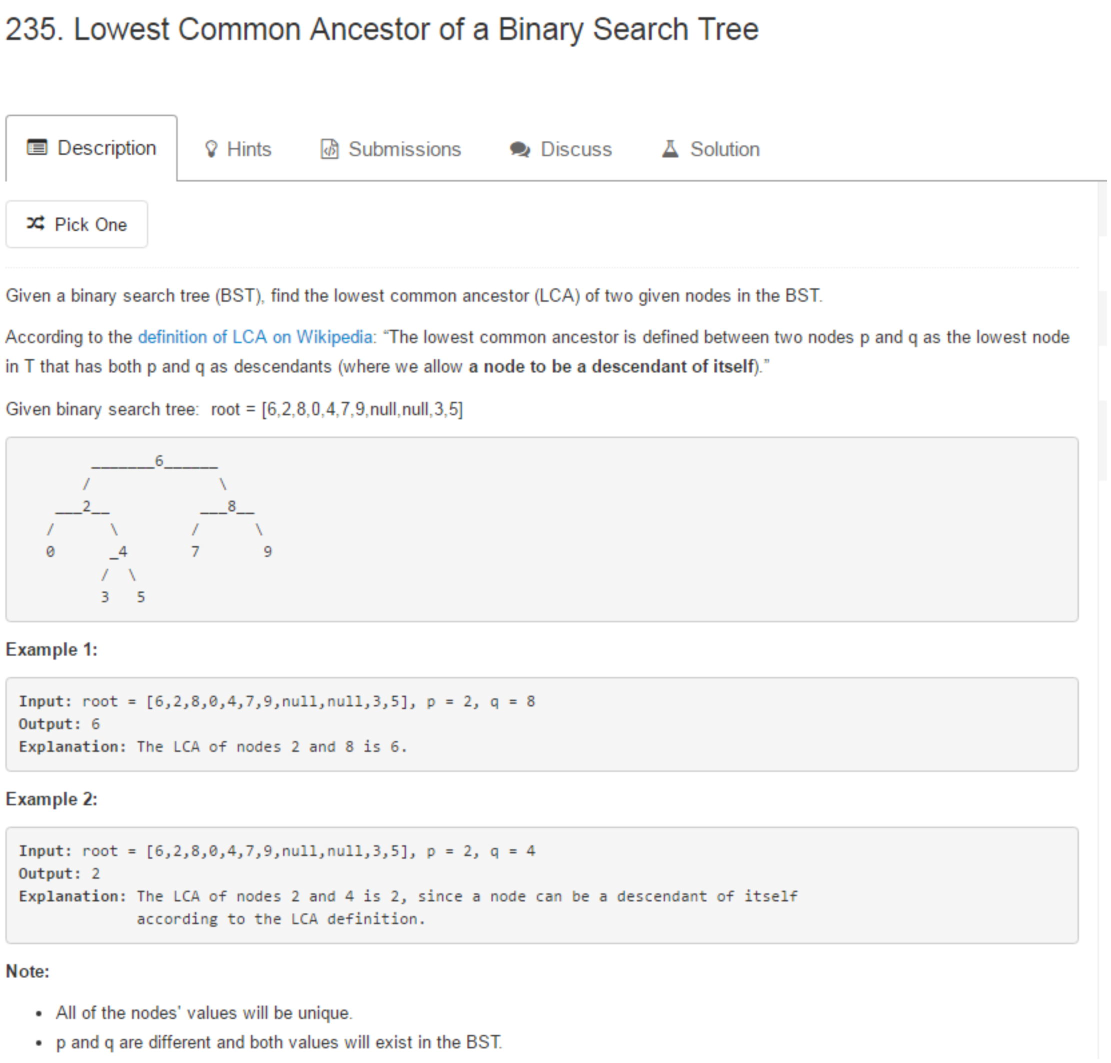


235. Lowest Common Ancestor of a Binary Search Tree

Description Hints Submissions Discuss Solution

Pick One

Given a binary search tree (BST), find the lowest common ancestor (LCA) of two given nodes in the BST.

According to the definition of LCA on Wikipedia: "The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow **a node to be a descendant of itself**)."

Given binary search tree: root = [6,2,8,0,4,7,9,null,null,3,5]

```
        _______6______
       /              \
    ___2__          ___8__
   /      \        /      \
   0      _4       7       9
         /  \
         3   5
```

Example 1:

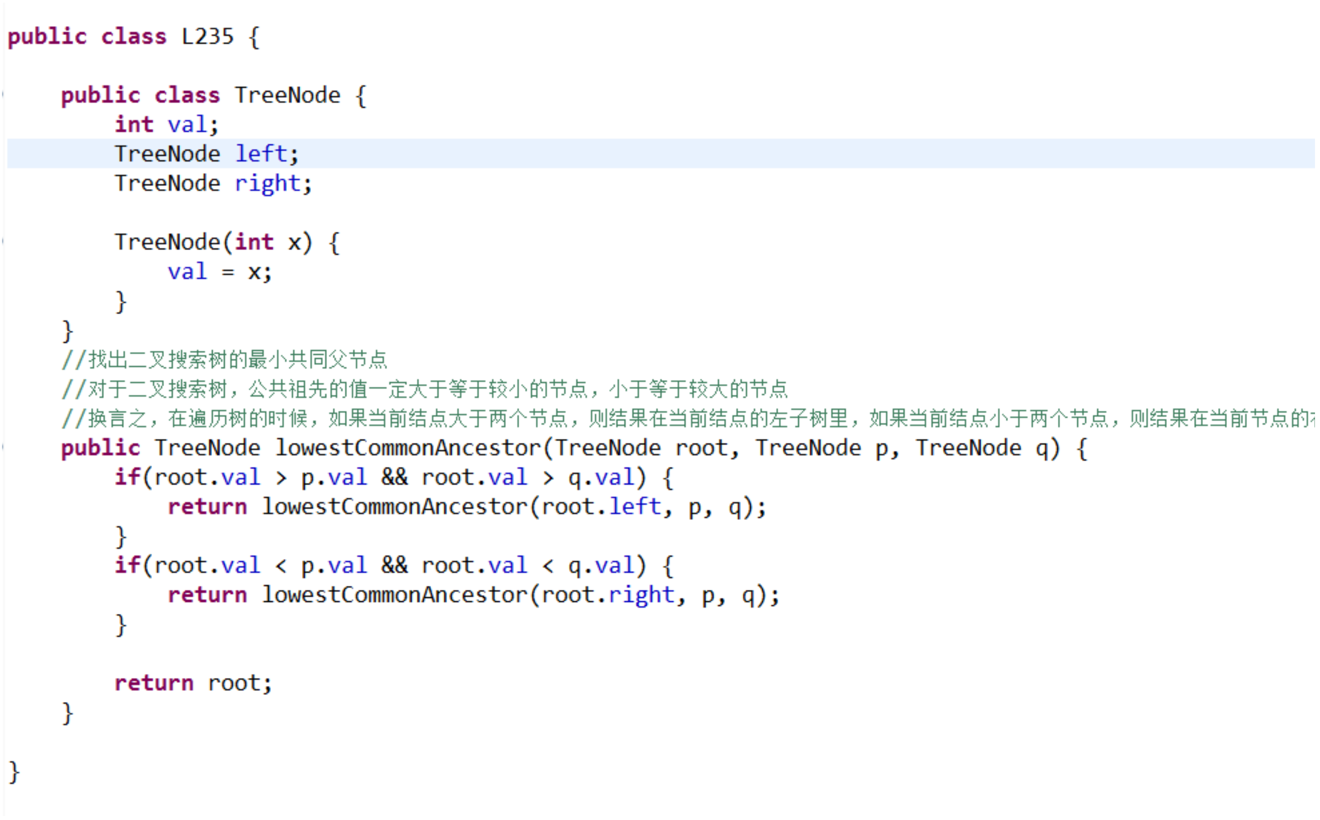
```
Input: root = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 8
Output: 6
Explanation: The LCA of nodes 2 and 8 is 6.
```

Example 2:

```
Input: root = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 4
Output: 2
Explanation: The LCA of nodes 2 and 4 is 2, since a node can be a descendant of itself
             according to the LCA definition.
```

Note:

- All of the nodes' values will be unique.
- p and q are different and both values will exist in the BST.



```java
public class L235 {

    public class TreeNode {
        int val;
        TreeNode left;
        TreeNode right;

        TreeNode(int x) {
            val = x;
        }
    }
    //找出二叉搜索树的最小共同父节点
    //对于二叉搜索树，公共祖先的值一定大于等于较小的节点，小于等于较大的节点
    //换言之，在遍历树的时候，如果当前结点大于两个节点，则结果在当前结点的左子树里，如果当前结点小于两个节点，则结果在当前节点的
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
        if(root.val > p.val && root.val > q.val) {
            return lowestCommonAncestor(root.left, p, q);
        }
        if(root.val < p.val && root.val < q.val) {
            return lowestCommonAncestor(root.right, p, q);
        }

        return root;
    }

}
```