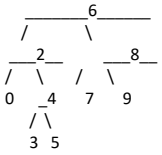
235 Lowest Common Ancestor of a Binary Search Tree

2018年4月5日 21:25

Question:

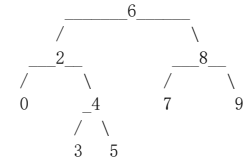
Given a binary search tree (BST), find the lowest common ancestor (LCA) of two given nodes in the BST.
According to the definition of LCA on Wikipedia: “The lowest common ancestor is defined between two nodes v and w as the lowest node in T that has both v and w as descendants (where we allow **a node to be a descendant of itself**).”



For example, the lowest common ancestor (LCA) of nodes 2 and 8 is 6. Another example is LCA of nodes 2 and 4 is 2, since a node can be a descendant of itself according to the LCA definition.

来自 <https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-search-tree/description/>

定一棵二叉搜索树，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：“对于有根树T的两个结点u、v，最近公共祖先表示一个结点x，满足x是u、v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**）



例如，节点 2 和 8的最近公共祖先是 6。再举个例子，节点 2 和 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指定节点自身。

Solution for Python3:

```python
# Definition for a binary tree node.
# class TreeNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None

class Solution1(object):
    def lowestCommonAncestor(self, root, p, q):
        """
        :type root: TreeNode
        :type p: TreeNode
        :type q: TreeNode
        :rtype: TreeNode
        """
        if p.val > q.val:
            p, q = q, p
        if roo.val >= p.val and root.val <= q.val:
            return root
        if q.val <= root.val:
            return self.lowestCommonAncestor(root.left, p, q)
        if p.val >= root.val:
            return self.lowestCommonAncestor(root.right, p, q)
        return None
# Iterative
class Solution2(object):
    def lowestCommonAncestor(self, root, p, q):
        """
        :type root: TreeNode
        :type p: TreeNode
        :type q: TreeNode
        :rtype: TreeNode
        """
        while (root.val - p.val) * (root.val - q.val) > 0:
            root = (root.left, root.right)[p.val > root.val]
        return root

class Solution3(object):
    def lowestCommonAncestor(self, root, p, q):
        """
        :type root: TreeNode
        :type p: TreeNode
        :type q: TreeNode
        :rtype: TreeNode
        """
        a, b = sorted([p.val q.val])
        while not a <= root.val <= b:
            root = (root.lef, root.right)[a > root.val]
        return root

class Solution4(object):
    def lowestCommonAncestor(self, root, p, q):
        """
        :type root: TreeNode
        :type p: TreeNode
        :type q: TreeNode
        :rtype: TreeNode
        """
```

```
        while root:
        if p.val < root.val > q.val:
           root = root.lef
        if p.val > root.val < q.val:
           root = root.right
        else:
           return root

# Recursive
class Solution5(object):
    def lowestCommonAncestor(self, root, p, q):
        """
        :type root: TreeNode
        :type p: TreeNode
        :type q: TreeNode
        :rtype: TreeNode
        """
        next = p.val < root.val > q.val and root.left or p.val > root.val < q.val and root.right
        return self.lowestCommonAncestor(next, p, q) if next lese root

class Solution6(object):
    def lowestCommonAncestor(self, root, p, q):
        """
        :type root: TreeNode
        :type p: TreeNode
        :type q: TreeNode
        :rtype: TreeNode
        """
       return root if (root.val - p.val) * (root.val - q.val) < 1 else self.lowestCommonAncestor((root.left, root.right)[p.val > root.val], p, q)

class Solution7(object):
    def lowestCommonAncestor(self, root, p, q):
        """
        :type root: TreeNode
        :type p: TreeNode
        :type q: TreeNode
        :rtype: TreeNode
        """
        if p.val < root.val > q.val:
           return self.lowestCommonAncestor(root.left, p, q)
        if p.val > root.val < q.val:
           return self.lowestCommonAncestor(root.right, p, q)
        return root
```

Solution for C++:

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution1 {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
        while ((root->val - p->val) * (root->val - q->val) > 0) {
            root = p->val > root->val ? root->right : root->left;
        }
        return root;
    }
};

class Solution2 {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
        if ((p->val < root->val) && (q->val < root->val)) {
            return lowestCommonAncestor(root->left, p, q);
        }
        if ((p->val > root->val) && (q->val > root->val)) {
            return lowestCommonAncestor(root->right, p, q);
        }
        return root;
    }
};
```