

## ★ 235 Lowest Common Ancestor of a Binary Search Tree

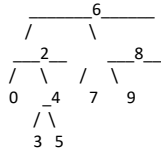
### Search Tree

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**).”

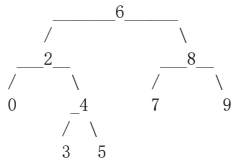


For example, the lowest common ancestor (LCA) of nodes 2 and 8 is 6. Another example is LCA of nodes 2 and 4 is 2, since a node can be a descendant of itself according to the LCA definition.

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:

```
1 # Definition for a binary tree node.
2 # class TreeNode(object):
3 #     def __init__(self, x):
4 #         self.val = x
5 #         self.left = None
6 #         self.right = None
7
8 class Solution1(object):
9     def lowestCommonAncestor(self, root, p, q):
10         """
11         :type root: TreeNode
12         :type p: TreeNode
13         :type q: TreeNode
14         :rtype: TreeNode
15         """
16         if p.val > q.val:
17             p, q = q, p
18         if root.val >= p.val and root.val <= q.val:
19             return root
20         if q.val <= root.val:
21             return self.lowestCommonAncestor(root.left, p, q)
22         if p.val >= root.val:
23             return self.lowestCommonAncestor(root.right, p, q)
24         return None
25
26 # Iterative
27 class Solution2(object):
28     def lowestCommonAncestor(self, root, p, q):
29         """
30         :type root: TreeNode
31         :type p: TreeNode
32         :type q: TreeNode
33         :rtype: TreeNode
34         """
35         while (root.val - p.val) * (root.val - q.val) > 0:
36             root = (root.left, root.right)[p.val > root.val]
37         return root
38
39 class Solution3(object):
40     def lowestCommonAncestor(self, root, p, q):
41         """
42         :type root: TreeNode
43         :type p: TreeNode
44         :type q: TreeNode
45         :rtype: TreeNode
46         """
47         a, b = sorted([p.val, q.val])
48         while not a <= root.val <= b:
49             root = (root.left, root.right)[a > root.val]
50         return root
51
52 class Solution4(object):
53     def lowestCommonAncestor(self, root, p, q):
54         """
55         :type root: TreeNode
56         :type p: TreeNode
57         :type q: TreeNode
58         :rtype: TreeNode
59         """
```

```

59     while root:
60         if p.val < root.val > q.val:
61             root = root.left
62         if p.val > root.val < q.val:
63             root = root.right
64         else:
65             return root
66
67 # Recursive
68 class Solution5(object):
69     def lowestCommonAncestor(self, root, p, q):
70         """
71         :type root: TreeNode
72         :type p: TreeNode
73         :type q: TreeNode
74         :rtype: TreeNode
75         """
76         next = p.val < root.val > q.val and root.left or p.val > root.val < q.val and root.right
77         return self.lowestCommonAncestor(next, p, q) if next else root
78
79 class Solution6(object):
80     def lowestCommonAncestor(self, root, p, q):
81         """
82         :type root: TreeNode
83         :type p: TreeNode
84         :type q: TreeNode
85         :rtype: TreeNode
86         """
87         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)
88
89 class Solution7(object):
90     def lowestCommonAncestor(self, root, p, q):
91         """
92         :type root: TreeNode
93         :type p: TreeNode
94         :type q: TreeNode
95         :rtype: TreeNode
96         """
97         if p.val < root.val > q.val:
98             return self.lowestCommonAncestor(root.left, p, q)
99         if p.val > root.val < q.val:
100             return self.lowestCommonAncestor(root.right, p, q)
101         return root

```

## Solution for C++:

```

1  /**
2   * Definition for a binary tree node.
3   * struct TreeNode {
4   *     int val;
5   *     TreeNode *left;
6   *     TreeNode *right;
7   *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
8   * };
9   */
10 class Solution1 {
11 public:
12     TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
13         while ((root->val - p->val) * (root->val - q->val) > 0) {
14             root = p->val > root->val ? root->right : root->left;
15         }
16         return root;
17     }
18 };
19
20 class Solution2 {
21 public:
22     TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
23         if ((p->val < root->val) && (q->val < root->val)) {
24             return lowestCommonAncestor(root->left, p, q);
25         }
26         if ((p->val > root->val) && (q->val > root->val)) {
27             return lowestCommonAncestor(root->right, p, q);
28         }
29         return root;
30     }
31 };

```