

```

1  /*
2   给定一个二叉搜索树，找到该树中两个指定节点的最近公共祖先。
3
4   百度百科中最近公共祖先的定义为：“对于有根树  $T$  的两个结点  $p$ 、 $q$ ，最近公共祖先表示为一个结点  $x$ ，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）。”
5
6   例如，给定如下二叉搜索树：  $root = [6,2,8,0,4,7,9,null,null,3,5]$ 
7
8
9
10  示例 1:
11
12  输入： $root = [6,2,8,0,4,7,9,null,null,3,5]$ ,  $p = 2$ ,  $q = 8$ 
13  输出：6
14  解释：节点 2 和节点 8 的最近公共祖先是 6。
15
16  示例 2:
17
18  输入： $root = [6,2,8,0,4,7,9,null,null,3,5]$ ,  $p = 2$ ,  $q = 4$ 
19  输出：2
20  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。
21
22
23
24  说明：
25
26      所有节点的值都是唯一的。
27       $p$ 、 $q$  为不同节点且均存在于给定的二叉搜索树中。
28
29  来源：力扣（LeetCode）
30  链接：https://leetcode-cn.com/problems/lowest-common-ancestor-of-a-binary-search-tree
31  著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
32  */

```

分析:

- 若 p, q 的 val 都小于当前的节点的 val ,则最近公共祖先一定出现在左子树;
- 若 p, q 的 val 都大于当前的节点的 val ,则最近公共祖先一定出现在右子树;
- 其他情况下,当前节点就是最近公共祖先.

方法一:C++_左右根遍历

```

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 Solution
11     {
12     private:
13
14         void helper (   TreeNode*& ret_val      ,
15                       TreeNode*   node          ,
16                       TreeNode*   p              ,
17                       TreeNode*   q              ,
18                       TreeNode*   q
19         )
20         {
21             if(node==NULL)
22             {
23                 return;
24             }
25
26             if(node->val < p->val && node->val < q->val)
27             {
28                 helper(ret_val,node->right,p,q);
29                 return;
30             }
31
32             if(node->val > p->val && node->val > q->val)
33             {
34                 helper(ret_val,node->left,p,q);
35                 return;
36             }
37             ret_val = node;
38             return;
39
40
41
42
43
44         }
45
46     public:
47         TreeNode* lowestCommonAncestor( TreeNode*   root      ,
48                                         TreeNode*   p        ,
49                                         TreeNode*   q        ,
50                                         TreeNode*   q
51                                         )
52         {
53             TreeNode* ret_val = NULL;
54             helper(ret_val,root,p,q);
55             return ret_val;
56
57         }
58     };
59
60
61     /*
62     执行结果:
63     通过
64     显示详情
65     执行用时 :40 ms, 在所有 cpp 提交中击败了82.59% 的用户

```

66 内存消耗 :25.5 MB, 在所有 cpp 提交中击败了96.73%的用户
67 */

AlimyBreak
2019.12.07