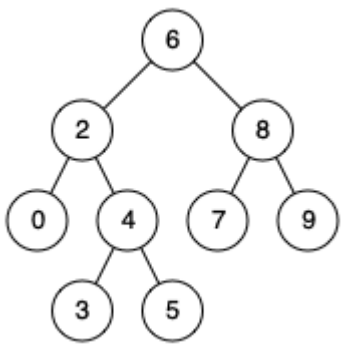
235. Lowest Common Ancestor of a Binary Search Tree

Easy 6488 207 Add to List Share

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

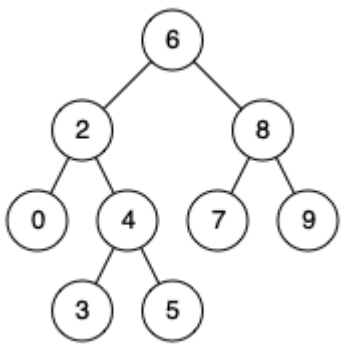
According to the definition of LCA on Wikipedia: “The lowest common ancestor is defined between two nodes `p` and `q` as the lowest node in `T` that has both `p` and `q` as descendants (where we allow **a node to be a descendant of itself**).”

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.

Example 3:

Input: root = [2,1], p = 2, q = 1
Output: 2

Constraints:

- The number of nodes in the tree is in the range `[2, 10^5]`.
- `-10^9 <= Node.val <= 10^9`
- All `Node.val` are **unique**.
- `p != q`
- `p` and `q` will exist in the BST.

Accepted 848,204 Submissions 1,457,941

Seen this question in a real interview before? Yes No

Companies

C++ Autocomplete

```cpp
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */

class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
        if(root == nullptr)
            return root;
        TreeNode* l = lowestCommonAncestor(root->left, p, q);
        TreeNode* r = lowestCommonAncestor(root->right, p, q);

        if(root == p || root == q || (l != nullptr and r != nullptr))
            return root;

        return l == nullptr? r: l;
    }
};
```

Testcase Run Code Result Debugger

Accepted Runtime: 4 ms

Your input: [6,2,8,0,4,7,9,null,null,3,5] 2

Output: 6 Diff

Expected: 6