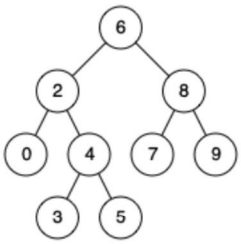
235. Lowest Common Ancestor of a Binary Search Tree

02 April 2022 08:58 PM

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

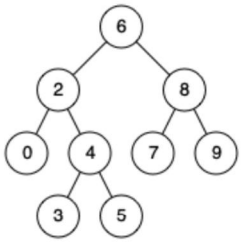
According to the definition of LCA on Wikipedia: "The lowest common ancestor is defined between two nodes `p` and `q` as the lowest node in `T` that has both `p` and `q` as descendants (where we allow **a node to be a descendant of itself**)."

Example 1:

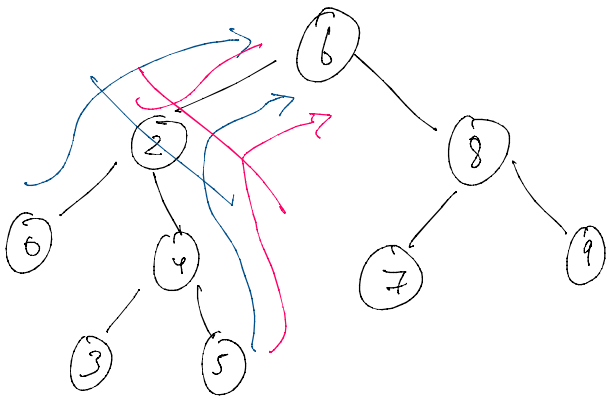



```
Input: root =
[6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 8
Output: 6
Explanation: The LCA of nodes 2 and 8 is 6.
```

Example 2:

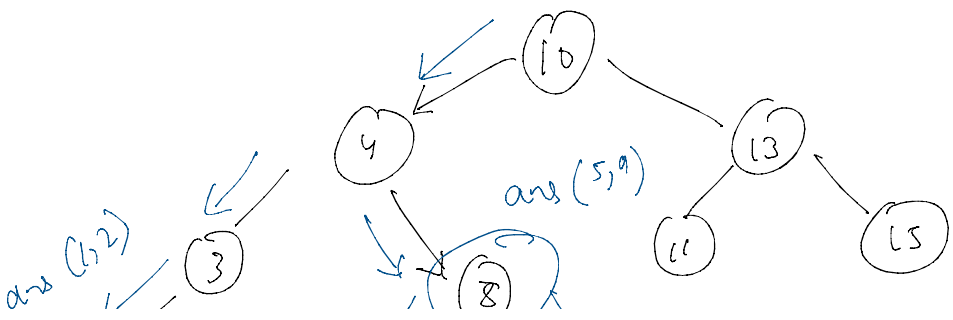


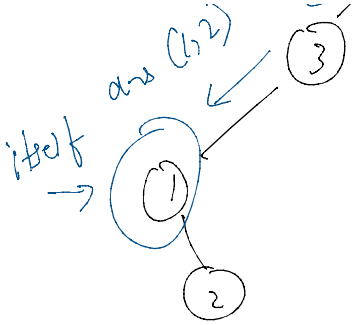

```
Input: root =
[6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 4
Output: 2
Explanation: The LCA of nodes 2 and 4 is 2,
since a node can be a descendant of itself
according to the LCA definition.
```

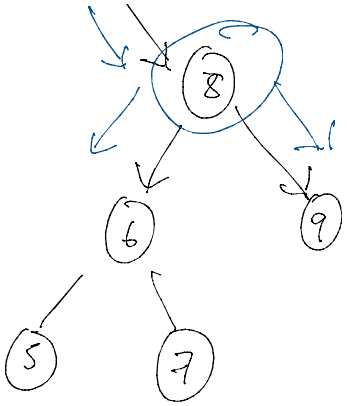


LCA (5, 0) ⇒ Intersection of (5, 0) is 2

LCA (2, 5) ⇒ Intersection of (2, 5) is 2





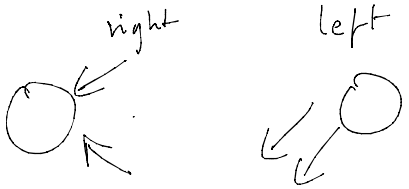


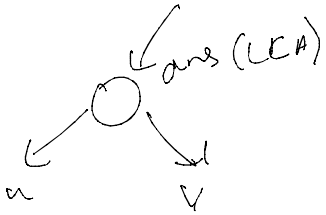
11 15

The point where you split is LCA.

LCA of (5,9) ⇒ 8 — possibility of two node

LCA of (1,2) ⇒ 1





T.C ⇒ O(H) → height of tree

S.C ⇒ O(1)

```
class Solution {
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
        if(root == null) return null;

        int curr = root.val;
        // both of them in right side
        if(curr < p.val && curr < q.val){
            return lowestCommonAncestor(root.right, p, q);
        }
        // both of them in left side
        if(curr > p.val && curr > q.val){
            return lowestCommonAncestor(root.left, p, q);
        }

        // if thts not the case then its the first point or last point of intersection
        return root;
    }
}
```