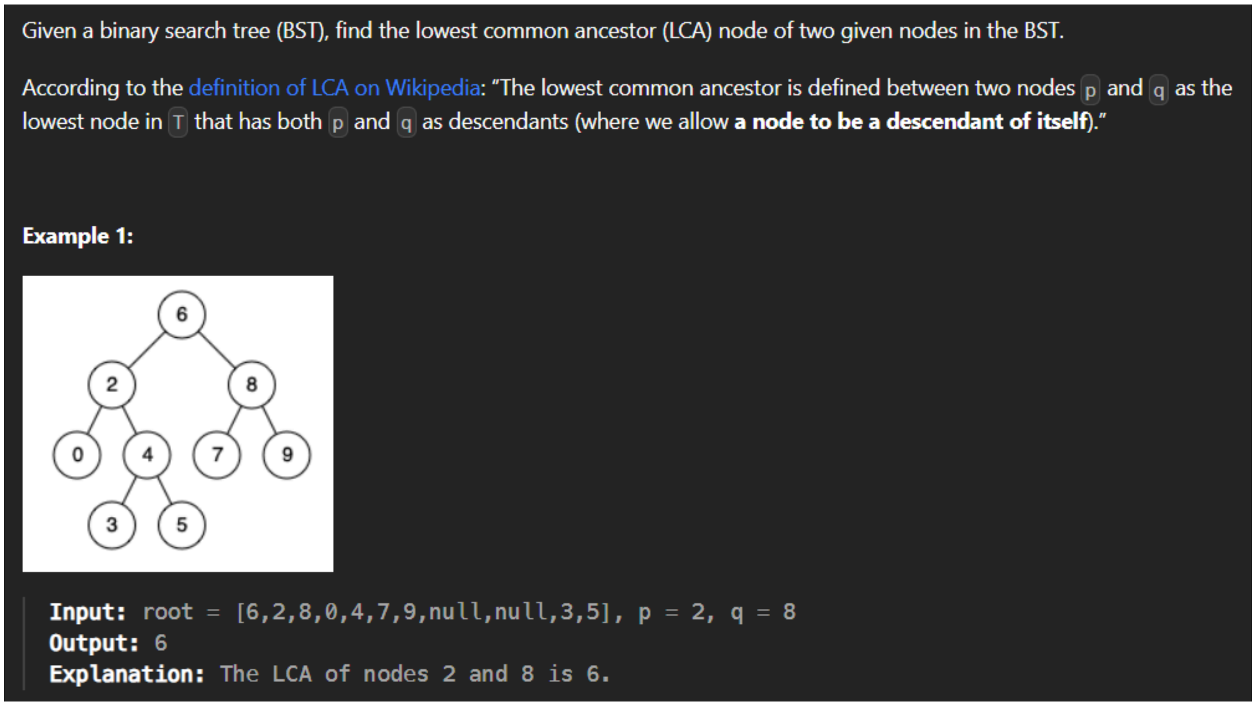
235. Lowest Common Ancestor of a Binary Search Tree

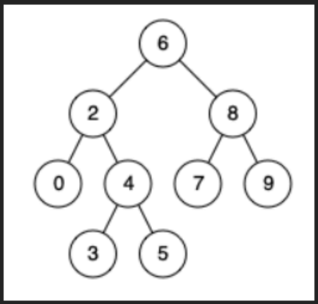
Problem Statement

1. 

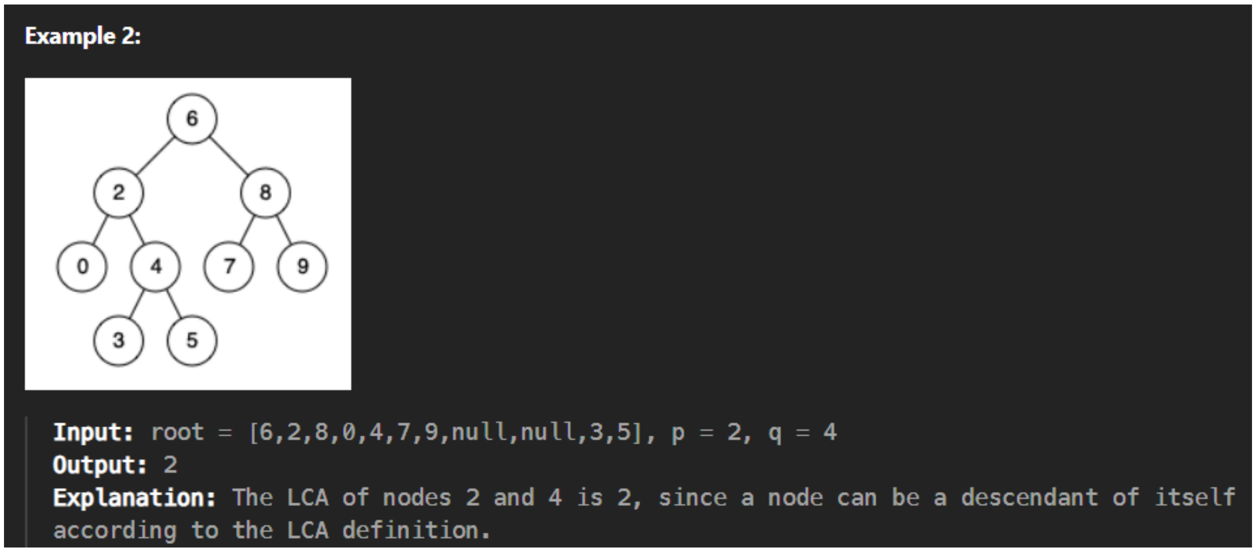
Given a binary search tree (BST), find the lowest common ancestor (LCA) node of two given nodes in the BST.

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**)."

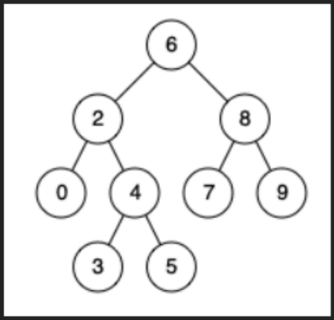
Example 1:



```
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.
```



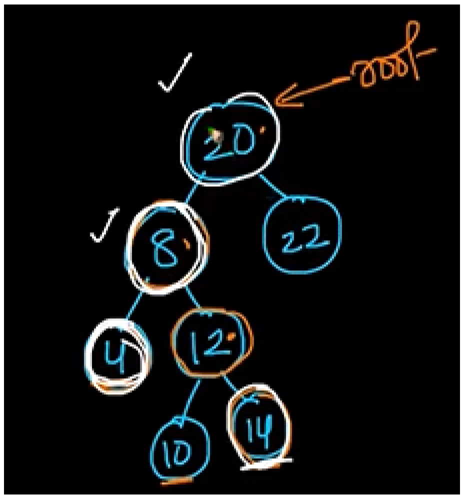
Example 2:

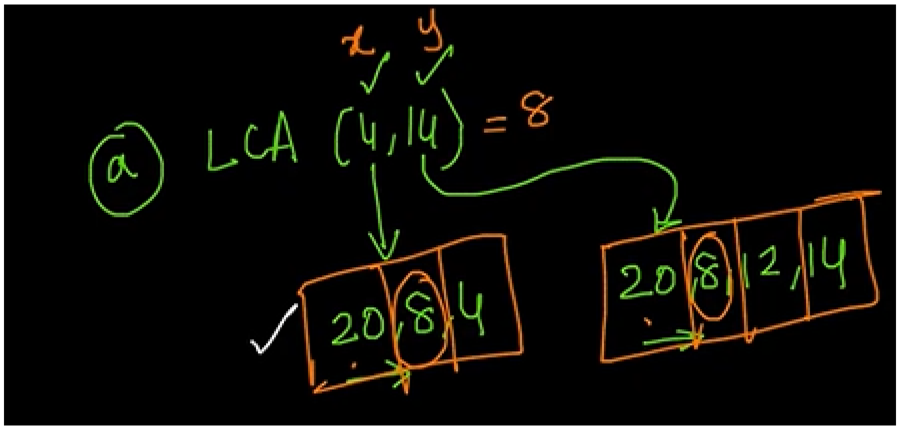


```
Input: root = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 4
Output: 2
Explanation: The LCA of nodes 2 and 4 is 2, since a node can be a descendant of itself
according to the LCA definition.
```

Approach:

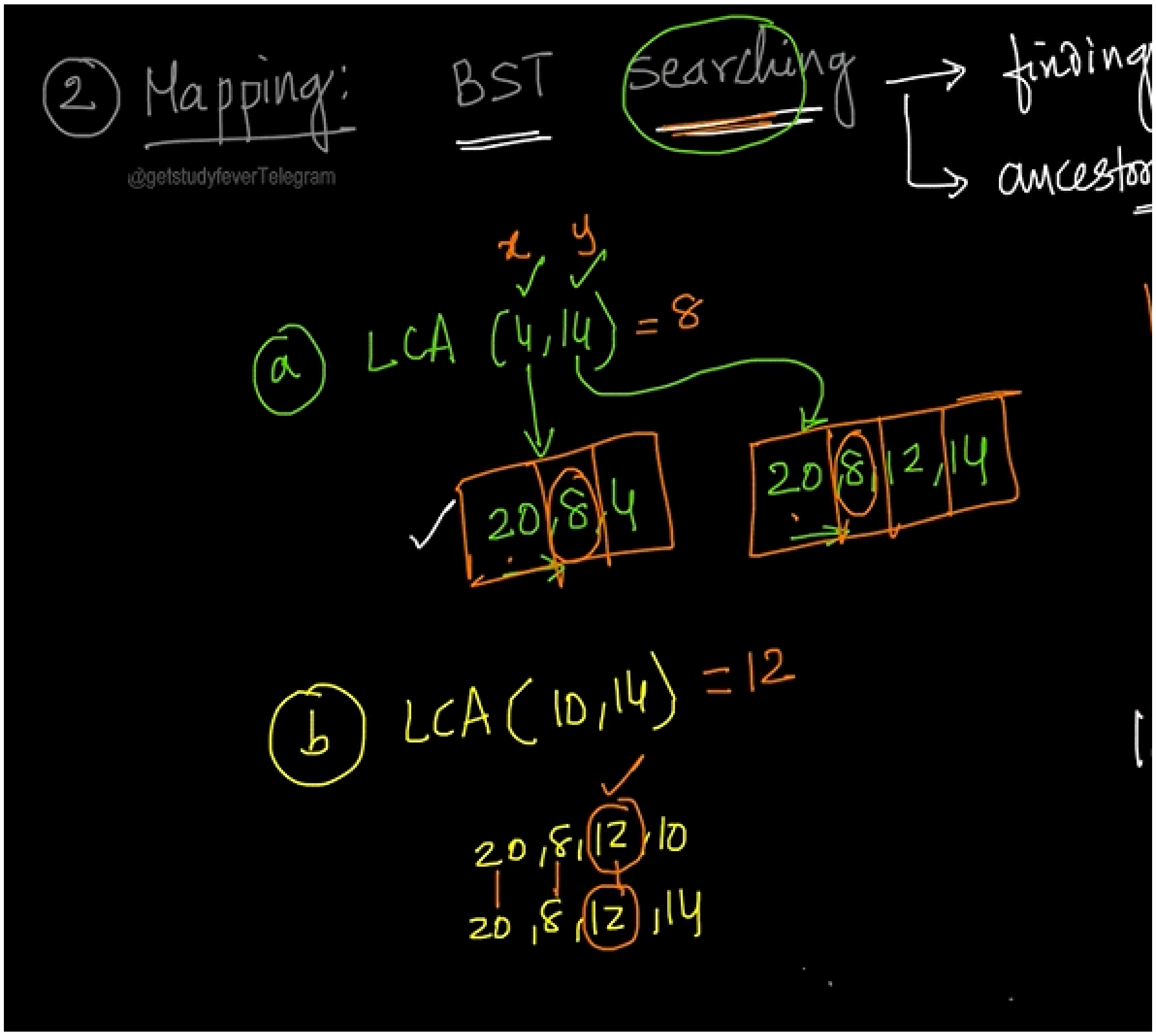
1. Here first approach would be to search p and storing its path in an array..and also search q and store its path in another arr

2. 



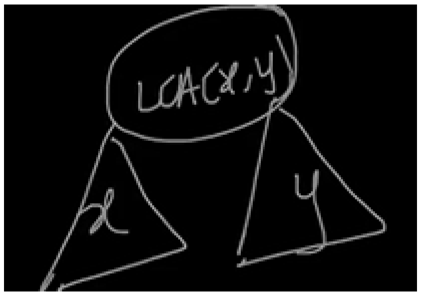
Now if we can

see…the last common elements is our LCA

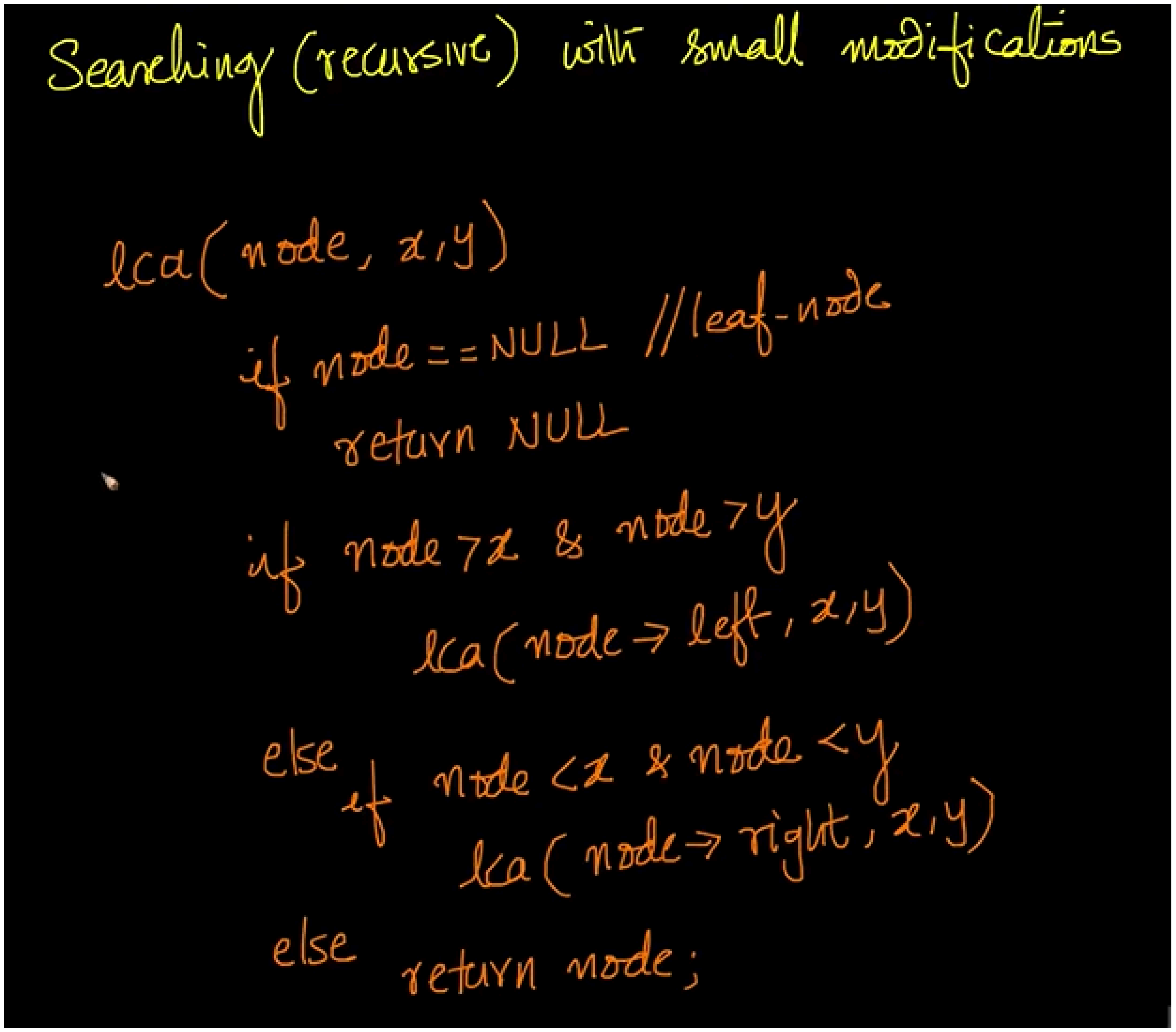


3. But here we are consuming extra space…can we solve this without using extra space?

2nd Approach using Tail Recursion

1. Here the main idea is that..LCA of x and y would be 

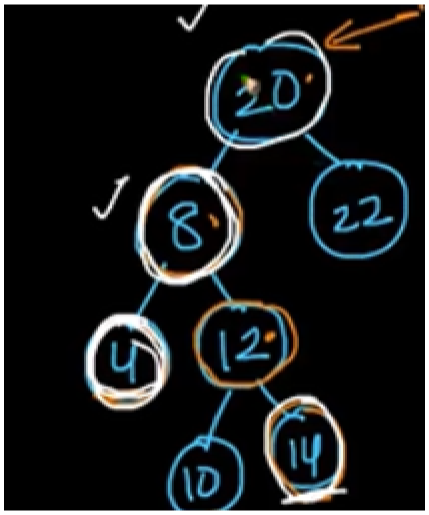
2. Refer the code once



Searching (recursive) with small modifications

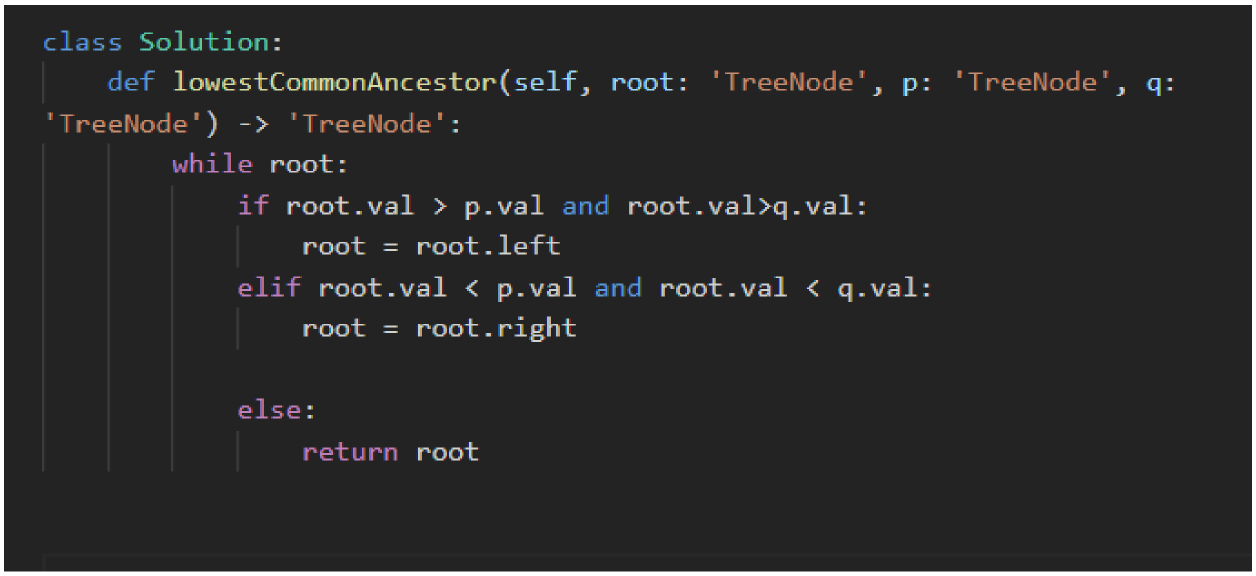
```
lca(node, x,y)
    if node==NULL //leaf-node
        return NULL
    if node>x & node>y
        lca(node->left, x,y)
    else if node<x & node<y
        lca(node->right, x,y)
    else return node;
```

3. Here if the node is null..then its the leaf node
4. And Node is greater than both x and y…then LCA would be found in node.left
5. Similarly if node is lesser than x and y…then LCA would be found in node.right
6. ELse we return the node..which will be our LCA
7. Here one might argue…that internally our recursion uses stack and consumes space…but here if we observe this is a tail recursion…where we dont need to go backward

8. For example… 

first we check with root node..and then next either we go left or right based on x&y…assume we went left..now for (4,14) …both the if and elif fails…and we return the node
9. So here we did not used stack and go back to root….so space complexity is O(n)

Python code:

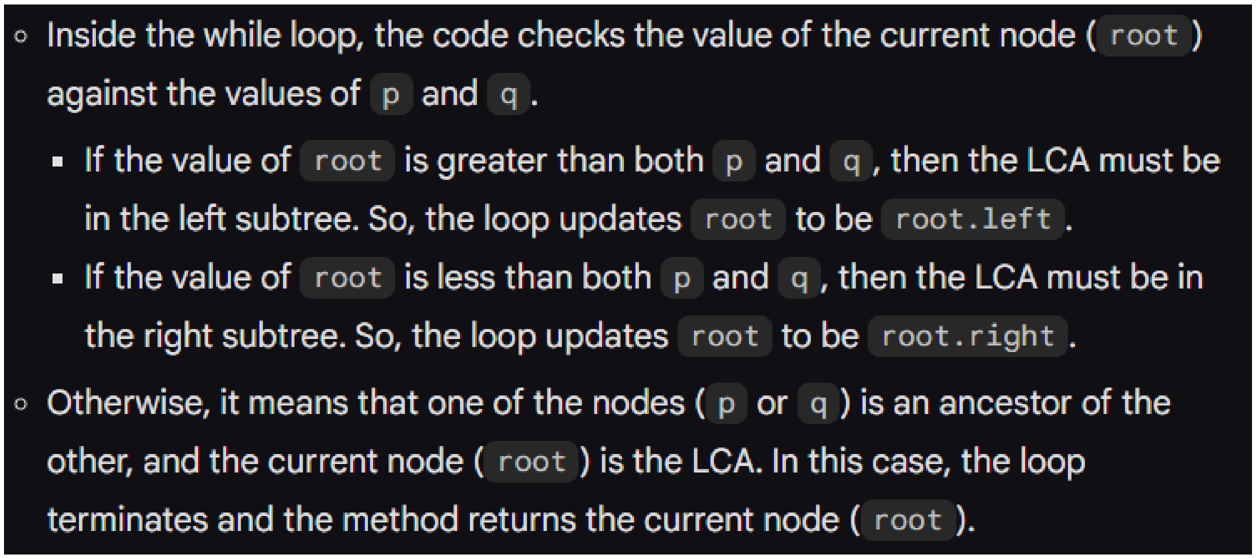
1. 

```
class Solution:
    def lowestCommonAncestor(self, root: 'TreeNode', p: 'TreeNode', q: 'TreeNode') -> 'TreeNode':
        while root:
            if root.val > p.val and root.val>q.val:
                root = root.left
            elif root.val < p.val and root.val < q.val:
                root = root.right

            else:
                return root
```

Explanation:

1.



- Inside the while loop, the code checks the value of the current node (`root`) against the values of `p` and `q`.
  - If the value of `root` is greater than both `p` and `q`, then the LCA must be in the left subtree. So, the loop updates `root` to be `root.left`.
  - If the value of `root` is less than both `p` and `q`, then the LCA must be in the right subtree. So, the loop updates `root` to be `root.right`.
- Otherwise, it means that one of the nodes (`p` or `q`) is an ancestor of the other, and the current node (`root`) is the LCA. In this case, the loop terminates and the method returns the current node (`root`).