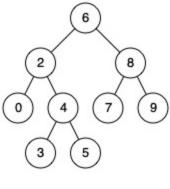
### 235. Lowest Common Ancestor of a Binary Search Tree

Dec. 16, 2018 | 87K views

Average Rating: 4.83 (78 votes)

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)." Given binary search tree: root = [6,2,8,0,4,7,9,null,null,3,5]



## **Input:** root = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 8

Example 1:

```
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.
Constraints:
```

All of the nodes' values will be unique.

p and q are different and both values will exist in the BST.

## But, binary search tree's property could be utilized, to come up with a better algorithm.

Solution

Lets review properties of a BST:

We can solve this using the approaches to find LCA in a binary tree.

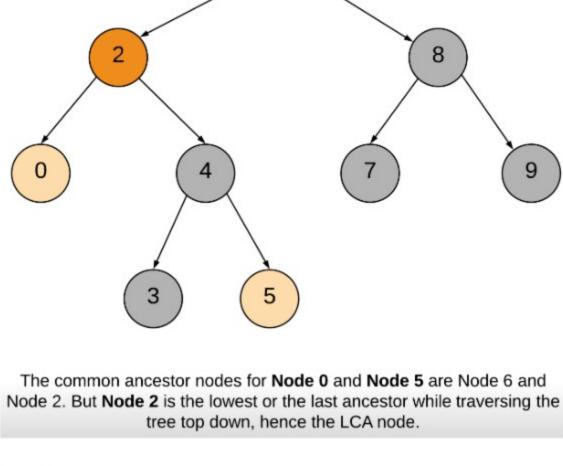
1. Left subtree of a node N contains nodes whose values are lesser than or equal to node N's

2. Right subtree of a node N contains nodes whose values are greater than node N's value. 3. Both left and right subtrees are also BSTs.

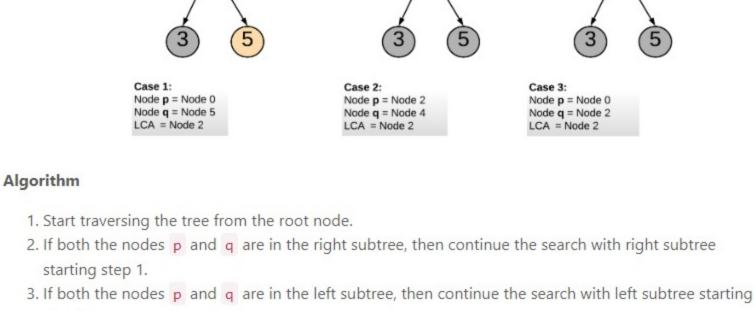
Approach 1: Recursive Approach Lowest common ancestor for two nodes p and q would be the last ancestor node common to both of them. Here last is defined in terms of the depth of the node. The below diagram would help in understanding what lowest means.

Following cases are possible:

Intuition



Note: One of p or q would be in the left subtree and the other in the right subtree of the LCA node.



### 4. If both step 2 and step 3 are not true, this means we have found the node which is common to node p 's and q 's subtrees. and hence we return this common node as the LCA.

8 9

17 18

19

20

21

22

23

24

25 26

Algorithm

Java

4

8 9 10

11

12 13

14

Python

1 class Solution:

step 1.

Python

# Value of current node or parent node.

# If both p and q are greater than parent

# If both p and q are lesser than parent

if p\_val > parent\_val and q\_val > parent\_val:

elif p\_val < parent\_val and q\_val < parent\_val:

# We have found the split point, i.e. the LCA node.

stack would be N since the height of a skewed BST could be N.

return self.lowestCommonAncestor(root.right, p, q)

return self.lowestCommonAncestor(root.left, p, q)

:rtype: TreeNode

- **Сору** Java 1 class Solution: def lowestCommonAncestor(self, root, p, q):
- 3 4 :type root: TreeNode 5 :type p: TreeNode :type q: TreeNode
- 10 parent\_val = root.val 11 12 # Value of p 13 p\_val = p.val 14 15 # Value of q 16  $q_val = q.val$

# **Complexity Analysis** ullet Time Complexity: O(N), where N is the number of nodes in the BST. In the worst case we might be visiting all the nodes of the BST. • Space Complexity: O(N). This is because the maximum amount of space utilized by the recursion

def lowestCommonAncestor(self, root, p, q):

:type root: TreeNode :type p: TreeNode :type q: TreeNode :rtype: TreeNode

# Value of p

# Value of q

q\_val = q.val

p\_val = p.val

return root

Approach 2: Iterative Approach

the split point. The point from where p and q won't be part of the same subtree or when one is the parent of the other.

**Сору** 

Post

A Report

function, we traverse down the tree iteratively. This is possible without using a stack or recursion since we don't need to backtrace to find the LCA node. In essence of it the problem is iterative, it just wants us to find

The steps taken are also similar to approach 1. The only difference is instead of recursively calling the

15 # Start from the root node of the tree 16 17 node = root 18 19 # Traverse the tree 20 while node: 21 22 # Value of current node or parent node. 23 parent\_val = node.val 24 25 if p\_val > parent\_val and q\_val > parent\_val: # If both p and q are greater than parent 26 node = node.right **Complexity Analysis** ullet Time Complexity : O(N), where N is the number of nodes in the BST. In the worst case we might be visiting all the nodes of the BST. • Space Complexity : O(1). Rate this article: \* \* \* \* \* O Previous Next **0** Comments: 28 Sort By -

Type comment here... (Markdown is supported)

Preview

something like this ....

SHOW 2 REPLIES

(123)

0 ∧ ∨ Ø Share ¬ Reply

Very consice explain and code. 0 A V & Share Reply

btn \* binarytree::findyoungestparent(btn\* c,int small, int big)

Read More

For the time complexity, I agree it should be O(n) considering the worst case when all node only has one child and p,q are near the bottom. For a balanced BST, the time complexity may be O(log(n)) since we reduce the nodes to check by half after each step. 71 A V 🗗 Share 🦘 Reply **SHOW 6 REPLIES** user4695hU # 21 @ December 19, 2018 6:16 AM Great solution! 10 A V C Share Reply SHOW 1 REPLY arzgania \* 60 ② July 9, 2019 1:24 PM Isn't the recursive solution tail-recursive (since the last call in every stack frame always just a recursive call by itself)? So wouldn't the space complexity be O(1)? 8 A V C Share Share SHOW 2 REPLIES jitwit \* 33 ② July 9, 2019 10:01 AM Small nitpick, isn't it more accurate to call time complexity O(h), where h is the height of the tree? If the tree is totally unbalanced, sure, h=N, but in general that will not be the case! 6 A V C Share Share SHOW 3 REPLIES gpadmaku \* 4 ② December 17, 2018 6:46 AM This was really helpful. Thanks! 4 A V C Share Share SHOW 1 REPLY This algorithm works under the assumption that p and q are always present in the tree. It just compares the values is greater that or not. Anyone has an idea how to change the algorithm to consider the case when p or q not present then return null 2 A V Share Share Reply **SHOW 4 REPLIES** A Report both approaches are absolutely top notch. thanks! 1 A V Share Share Reply I'm kinda confused. The solution assumes that the Binary Tree is a BST but the problem statement does not state that the Binary Tree is a BST. SHOW 2 REPLIES vivekprashant 🛊 0 🗿 December 21, 2018 9:49 PM we dont need to compare both the values, instead figure out higher and lower of the input and if parent is lower than smaller value iterate right it we find a node with value higher than smaller value.