Solution Review: Find Ancestors of a given node in a BST

This review provides a detailed analysis of the different ways to solve the Find Ancestors of a given node in a Binary Tree challenge

We'll cover the following

Solution #1: Using a recursive helper function

main.py

BinarySearchTree.py

Node.py

```
from Node import Node
from BinarySearchTree import BinarySearchTree


def findAncestors(root, k):
    result = []
    recfindAncestors(root, k, result)  # recursively find ancestors
    return str(result)  # return a string of ancestors
```

```
    return str(result)  # return a string of ancestors
 9
10
11  def recfindAncestors(root, k, result):
12      if root is None:  # check if root exists
13          return False
14      elif root.val is k:  # check if val is k
15          return True
16      recur_left = recfindAncestors(root.leftChild, k, result)
17      recur_right = recfindAncestors(root.rightChild, k, result)
18      if recur_left or recur_right:
19          # if recursive find in either left or right sub tree
20          # append root value and return true
21          result.append(root.val)
22          return True
23      return False  # return false if all failed
24
25
26  BST = BinarySearchTree(6)
27  BST.insert(1)
28  BST.insert(133)
```

This solution uses a recursive helper function that The recursive function starts traversing from the root till the input node and backtracks to append the ancestors that led to the node.

Time Complexity#

This is an $O(n)$ time function since it iterates over all of the nodes of the entire tree.

Solution #2: Iteration#

main.py

BinarySearchTree.py

Node.py

```
from Node import Node
from BinarySearchTree import BinarySearchTree


def findAncestors(root, k):
    if not root:  # check if root exists
        return None
    ancestors = []  # empty list of ancestors
    current = root  # iterator current set to root

    while current is not None:  # iterate until we reach None
        if k > current.val:  # go right
            ancestors.append(current.val)
            current = current.rightChild
        elif k < current.val:  # go left
            ancestors.append(current.val)
            current = current.leftChild
        else:  # when k == current.val
            return ancestors[::-1]
    return []


BST = BinarySearchTree(6)
BST.insert(1)
BST.insert(133)
BST.insert(12)
print(findAncestors(BST.root, 12))
```

This solution conducts a search for `k` in the BST until a `None` node or k itself is reached. If `k` is reached, the ancestors are returned, otherwise, an empty list is returned.

Time Complexity#

The time complexity of this solution is $O(log(n))$ since a path from the root to `k` is traced.

Back

Challenge 3: Find Ancestors of a given...

Next

Challenge 4: Find the Height of a BST

Completed

Report an Issue