



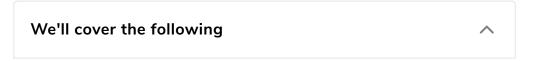






Solution Review: Find Nodes at "k" distance from the Root

This review provides a detailed analysis of the different ways to solve the Find Nodes at "k" distance from the Root challenge.



- Solution: Checking all Nodes at distance k Recursively
 - Time Complexity

Solution: Checking all Nodes at distance k Recursively

```
main.py
BinarySearchTree.py
Node.py
     from Node import Node
  2
     from BinarySearchTree import BinarySearchTree
  3
    def findKNodes(root, k):
  5
         res = []
  6
         findK(root, k, res) # recurse the tree for node at k distance
         return str(res)
  8
  9
 10
 11
     def findK(root, k, res):
```

```
12
        if root is None: # return if root does not exist
                                                             []
13
            return
14
        if k == 0:
            res.append(root.val) # append as root is kth node
15
16
        else:
17
            # check recursively in both sub-tree for kth node
            findK(root.leftChild, k - 1, res)
18
19
            findK(root.rightChild, k - 1, res)
20
21
22
   BST = BinarySearchTree(6)
   BST.insert(4)
23
24
   BST.insert(9)
25 BST.insert(5)
26 BST.insert(2)
27 BST.insert(8)
28 BST.insert(12)
```

This solution maintains a counter k that is decremented until it is 0 or a leaf node is reached, returning the nodes that are encountered at k == 0

Time Complexity#

The time complexity of this solution is in O(n)

Interviewing soon? We've partnered with Hired so that $$\times$$ companies apply to you instead of you applying to them. See how \odot



Challenge 5: Find Nodes at "k" distanc...



Trees Quiz: Test your understanding ot...



