
Average Rating: 3.88 (8 votes)

742. Closest Leaf in Binary Tree 💆

Dec. 10, 2017 | 13.3K views

Given a binary tree **where every node has a unique value**, and a target key **k**, find the value of the nearest leaf node to target **k** in the tree.

Here, nearest to a leaf means the least number of edges travelled on the binary tree to reach any leaf of the tree. Also, a node is called a *leaf* if it has no children.

In the following examples, the input tree is represented in flattened form row by row. The actual root tree given will be a TreeNode object.

Example 1:

Example 2:

```
Input:
root = [1], k = 1
Output: 1

Explanation: The nearest leaf node is the root node itself.
```

Example 3:

Note:

- root represents a binary tree with at least 1 node and at most 1000 nodes.
 Every node has a unique node.val in range [1, 1000].
- 3. There exists some node in the given binary tree for which node.val == k.

Approach #1: Convert to Graph [Accepted]

Intuition

using breadth-first search.

Algorithm

Instead of a binary tree, if we converted the tree to a general graph, we could find the shortest path to a leaf

We use a depth-first search to record in our graph each edge travelled from parent to node.

After, we use a breadth-first search on nodes that started with a value of ${\bf k}$, so that we are visiting nodes in

order of their distance to k. When the node is a leaf (it has one outgoing edge, where the root has a "ghost" edge to null), it must be the answer.

| Java | Python |

```
Java Python
  1 class Solution(object):
        def findClosestLeaf(self, root, k):
            graph = collections.defaultdict(list)
            def dfs(node, par = None):
                    graph[node].append(par)
                    graph[par].append(node)
                    dfs(node.left, node)
  9
                    dfs(node.right, node)
  10
            dfs(root)
  11
  12
            queue = collections.deque(node for node in graph
  13
                                   if node and node.val == k)
  14
            seen = set(queue)
  15
  16
            while queue:
  17
                node = queue.popleft()
               if node:
  18
               if len(graph[node]) <= 1:
  19
  20
  21
                   for nei in graph[node]:
  22
                      if nei not in seen:
  23
                           seen.add(nei)
                           queue.append(nei)
Complexity Analysis
```

Time Complexity: O(N) where N is the number of nodes in the given input tree. We visit every node a constant number of times.

- Space Complexity: O(N), the size of the graph.
- Approach #2: Annotate Closest Leaf [Accepted]

Intuition and Algorithm

Say from each node, we already knew where the closest leaf in it's subtree is. Using any kind of traversal plus memoization, we can remember this information.

Then the closest leaf to the target (in general, not just subtree) has to have a lowest common ancestor with the target that is on the path from the root to the target. We can find the path from root to

target via any kind of traversal, and look at our annotation for each node on this path to determine all leaf candidates, choosing the best one.

Dava Python

class Solution(object):

```
def findClosestLeaf(self, root, k):
            annotation = {}
             def closest leaf(root):
                 if root not in annotation:
                     if not root:
                         ans = float('inf'), None
                     elif not root.left and not root.right:
                        ans = 0, root
  10
                     else:
                         d1, leaf1 = closest_leaf(root.left)
  11
  12
                         d2, leaf2 = closest_leaf(root.right)
  13
                         ans = min(d1, d2) + 1, leaf1 if d1 < d2 else leaf2
                     annotation[root] = ans
  14
  15
                 return annotation[root]
  16
  17
             #Search for node.val == k
  18
             path = []
  19
             def dfs(node):
  20
                 if not node:
  21
                     return
                 if node.val == k:
  22
  23
                     path.append(node)
  24
                     return True
 25
                 path.append(node)
 26
                 ans1 = dfs(node.left)
                 if ans1: return True
 27
                 ans2 = dfs(node.right)
 28
Complexity Analysis
  • Time and Space Complexity: O(N). The analysis is the same as in Approach #1.
```

Time and Space Analysis written by

3 Previous

Rate this article: * * * * *

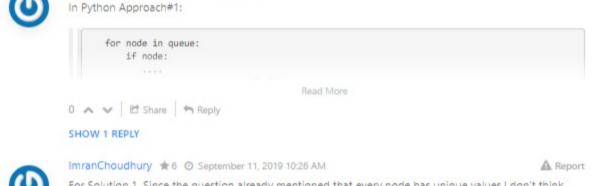
Analysis written by: @awice.

```
Comments: 8
                                                                                                     Sort By ▼
              Type comment here... (Markdown is supported)
              @ Preview
                                                                                                       Post
             coder001 ★18 ② November 15, 2018 11:36 PM
             In solution 1, condition if len(graph[node]) <= 1: can be simplified to if len(graph[node])
             == 1: as all leaf nodes have exactly only one neighbor.
             1 A V E Share A Reply
             SHOW 2 REPLIES
             rajatmittal18 * 225   June 28, 2019 8:30 AM
             In solution 2, we actually don't need to call closest_leaf function more than once, just call it once, form
             the annotation dictionary and use it while traversing the path.
             0 A V E Share A Reply
             kanerodriguezpro *1 ② May 19, 2019 8:28 PM
             Simple tree traversal beats 98% time | 67% space
              class Solution:
                  def findClosestLeaf(self, root: TreeNode, k: int) -> int:
                      def helper(node: TreeNode. k: int) -> (int.int.int):
                                                       Read More
             0 A V Et Share  Reply
```

Next 0



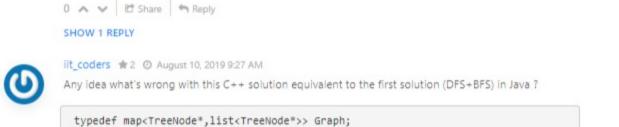




For Solution 1, Since the question already mentioned that every node has unique values,I don't think we need the seen = set(queue) here. We can simply use queue for the BFS.

graph = collections.defaultdict(list)
def dfs(node.par = None):

Read More



Read More

0 A V B Share A Reply

xivaxy ★0 ② December 12, 2017 11:39 PM