Description | Solution | Discuss (270) | Submissions

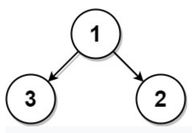
742. Closest Leaf in a Binary Tree

Medium 677 127 Add to List Share

Given the `root` of a binary tree where every node has **a unique value** and a target integer `k`, return *the value of the **nearest leaf node** to the target `k` in the tree.*

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

Example 1:

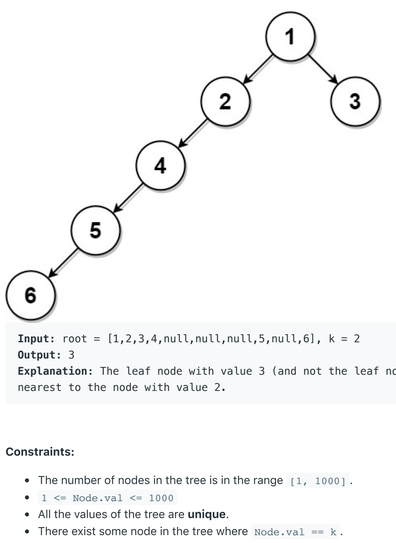



```
Input: root = [1,3,2], k = 1
Output: 2
Explanation: Either 2 or 3 is the nearest leaf node to the target of 1.
```

Example 2:


```
Input: root = [1], k = 1
Output: 1
Explanation: The nearest leaf node is the root node itself.
```

Example 3:




```
Input: root = [1,2,3,4,null,null,null,5,null,6], k = 2
Output: 3
Explanation: The leaf node with value 3 (and not the leaf node with value 6) is
nearest to the node with value 2.
```

Constraints:

- The number of nodes in the tree is in the range `[1, 1000]`.
- `1 <= Node.val <= 1000`
- All the values of the tree are **unique**.
- There exist some node in the tree where `Node.val == k`.

Accepted 33,252 | Submissions 74,056

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Facebook | 3

Related Topics

Tree Depth-First Search Breadth-First Search Binary Tree

Hide Hint 1

Convert the tree to a general graph, and do a breadth-first search. Alternatively, find the closest leaf for every node on the path from root to target.

Java | Autocomplete

```java
class Solution {
    public int findClosestLeaf(TreeNode root, int k) {
        Map<TreeNode, List<TreeNode>> graph = new HashMap();
        dfs(graph, root, null);

        Queue<TreeNode> queue = new LinkedList();
        Set<TreeNode> seen = new HashSet();

        for (TreeNode node: graph.keySet()) {
            if (node != null && node.val == k) {
                queue.add(node);
                seen.add(node);
            }
        }

        while (!queue.isEmpty()) {
            TreeNode node = queue.poll();
            if (node != null) {
                if (graph.get(node).size() <= 1)
                    return node.val;
                for (TreeNode nei: graph.get(node)) {
                    if (!seen.contains(nei)) {
                        seen.add(nei);
                        queue.add(nei);
                    }
                }
            }
        }
        throw null;
    }

    public void dfs(Map<TreeNode, List<TreeNode>> graph, TreeNode node, TreeNode parent) {
        if (node != null) {
            if (!graph.containsKey(node)) graph.put(node, new LinkedList<TreeNode>());
            if (!graph.containsKey(parent)) graph.put(parent, new LinkedList<TreeNode>());
            graph.get(node).add(parent);
            graph.get(parent).add(node);
            dfs(graph, node.left, node);
            dfs(graph, node.right, node);
        }
    }
}
```