

742. Closest Leaf in a Binary Tree

[Description](#)[Hints](#)[Submissions](#)[Discuss](#)[Solution](#)

- Difficulty:Medium
- Total Accepted:958
- Total Submissions:3.3K
- Contributor: [1337c0d3r](#)

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

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:

Input :

root = [1, 3, 2], $k = 1$

Diagram of binary tree:



Output: 2 (or 3)

Explanation: Either 2 or 3 is the closest leaf node to 1.

Example 2:

Input :

root = [1], $k = 1$

Output: 1

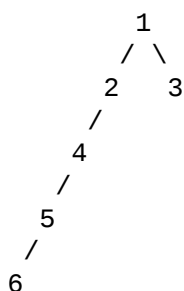
Explanation: The closest leaf node is the root node itself.

Example 3:

Input :

root = [1,2,3,4,null,null,null,5,null,6], $k = 2$

Diagram of binary tree:



Output: 3

Explanation: The leaf node with value 3 (and not the leaf node with value 6) is closest to the node with value 2.

Note:

1. `root` represents a binary tree with at least 1 node and at most 1000 nodes.
2. 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`.

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
    TreeNode closest = null;
    int minDist = Integer.MAX_VALUE;
    public int findClosestLeaf(TreeNode root, int k) {
        TreeNode[] ancestors = new TreeNode[1000];
        findClosestLeafRec(root, k, ancestors, 0);
        return closest==null?-1:closest.val;
    }
    private void findClosestLeafRec(TreeNode root, int k,
    TreeNode[] ancestors, int idx){
        if(root==null) return;
        else if(root.val==k){
            closestLeafBelow(root, 0);
            for(int i = idx-1; i >=0; i--){
                closestLeafBelow(ancestors[i], idx-i);
            }
            return;
        }
        ancestors[idx] = root;
        findClosestLeafRec(root.left, k, ancestors, idx+1);
        findClosestLeafRec(root.right, k, ancestors, idx+1);
    }
    private void closestLeafBelow(TreeNode root, int dist){
        if(root==null) return;
        else if(root.left==null && root.right==null){
            if(dist<minDist){
                minDist = dist;
                closest = root;
            }
        }
        return;
    }
}
```

```
        else{
            closestLeafBelow(root.left, dist+1);
            closestLeafBelow(root.right, dist+1);
        }
        return;
    }
}
```