Min distance between two given nodes of a Binary Tree

Given a binary tree with **n** nodes and two node values, **a** and **b**, your task is to find the minimum distance between them. The given two nodes are guaranteed to be in the binary tree and all node values are **unique**.

Example 1:

Input:

```
     1
    / \
   2   3
```

a = 2, b = 3

Output:
2

Explanation:

We need the distance between 2 and 3. Being at node 2, we need to take two steps ahead in order to reach node 3. The path followed will be: 2 -> 1 -> 3. Hence, the result is 2.

Example 2:

Input:

```
      11
     /  \
    22  33
   / \ / \
  44 55 66 77
```

a = 77, b = 22

Output:
3

Explanation:

We need the distance between 77 and 22. Being at node 77, we need to take three steps ahead in order to reach node 22. The path followed will be: 77 -> 33 -> 11 -> 22. Hence, the result is 3.

Your Task:
You don't need to read input or print anything. Your task is to complete the function **findDist()** which takes the **root** node of the tree and the two node values **a** and **b** as input parameters and returns the minimum distance between the nodes represented by the two given node values.

Expected Time Complexity: O(n).
Expected Auxiliary Space: O(Height of the Tree).

Constraints:
$2 <= n <= 10^5$
$0 <=$ Data of a node $<= 10^9$

```python
#User function Template for python3
'''
# Node Class:
class Node:
    def __init__(self,val):
        self.data = val
        self.left = None
        self.right = None
'''
class Solution:
    def findLCA(self, root, a, b):
        if root is None:
            return None

        if root.data == a or root.data == b:
            return root

        left_lca = self.findLCA(root.left, a, b)
        right_lca = self.findLCA(root.right, a, b)

        if left_lca and right_lca:
            return root

        return left_lca if left_lca is not None else right_lca
    # Function to find the distance from a given root to a
target node
    def findDistance(self, root, target, distance):
        if root is None:
            return -1

        if root.data == target:
            return distance

        left = self.findDistance(root.left, target, distance +
1)
        if left != -1:
            return left

        return self.findDistance(root.right, target, distance
+ 1)

    # Main function to find the minimum distance between nodes
a and b
    def findDist(self, root, a, b):
        # Step 1: Find the LCA of nodes a and b
        lca = self.findLCA(root, a, b)

        if lca is None:
            return -1  # If LCA is not found, return an error
code

        # Step 2: Find the distance from the LCA to node a
        d1 = self.findDistance(lca, a, 0)
        if d1 == -1:
            return -1  # If node a is not found, return an
error code

        # Step 3: Find the distance from the LCA to node b
        d2 = self.findDistance(lca, b, 0)
        if d2 == -1:
            return -1  # If node b is not found, return an
error code

        # Step 4: Return the sum of the distances
        return d1 + d2
```