Min distance between two given nodes of a Binary Tree

Given a binary tree and two node values your task is to find the minimum distance between them.

Input: 1
/
2 3
a = 2, b = 3 **Output:** 2 **Explanation:** The tree formed is:
1
/ \
2 3
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.

```
def findDist(root,a,b):
    if root:
        path1 = []
        path2 = []
        helper(root,a,path1)
        helper(root,b,path2)
        i = 0

        while i<len(path1) and i<len(path2):
            if path1[i]!=path2[i]:
                break
            i = i+1

        return (len(path1)+len(path2)-2*i)
    else:
        return 0

#Way to find path from root to a node
def helper(root,key,path):
    if root is None:
        return False
    path.append(root.data)
```

```
    if root.data == key:
        return True
    if helper(root.left, key,path) or helper(root.right, key,path):
        return True

    path.pop(-1)
    return False
```