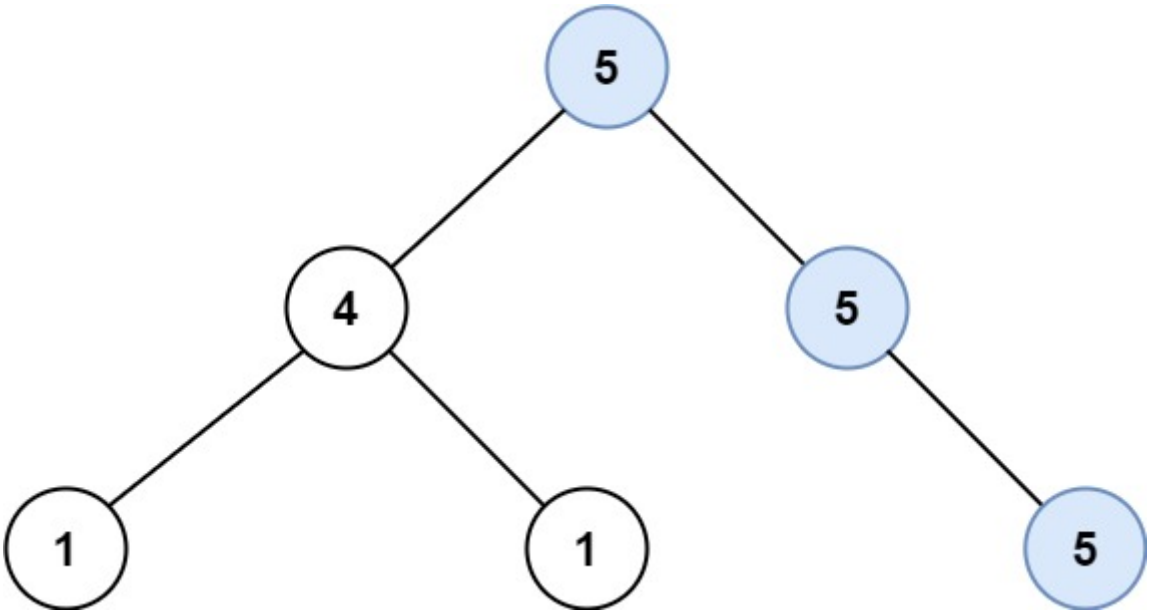
687. Longest Univalue Path

Given the `root` of a binary tree, return *the length of the longest path, where each node in the path has the same value*. This path may or may not pass through the root.

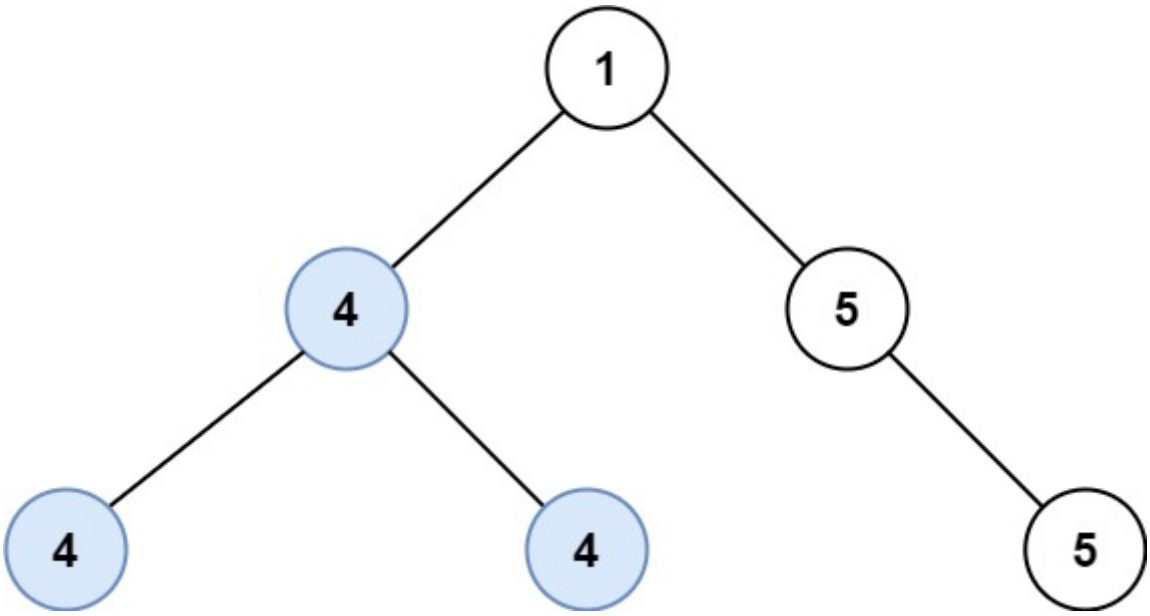
The length of the path between two nodes is represented by the number of edges between them.

Example 1:




```
Input: root = [5,4,5,1,1,5]
Output: 2
```

Example 2:



```
Input: root = [1,4,5,4,4,5]
Output: 2
```

Constraints:

- The number of nodes in the tree is in the range `[0, 10<sup>4</sup>]`.
- `-1000 <= Node.val <= 1000`
- The depth of the tree will not exceed `1000`.

```
import sys
class Pair:
    def __init__(self,val):
        self.univalue = val
        self.length = 0
class Solution:
    def longestUnivaluePath(self, root: Optional[TreeNode]) -> int:
        if root is None:
            return 0
        ans = [0]
        self.helper(root,ans)
        return ans[0]-1
    def helper(self,root,ans):
        if root is None:
            return Pair(sys.maxsize)
        left = self.helper(root.left,ans)
        right = self.helper(root.right,ans)
        myAns = Pair(root.val)
        #case-I
        if root.val == left.univalue and root.val==right.univalue:
            ans[0] = max(ans[0],left.length+right.length+1)
            myAns.length = max(left.length,right.length)+1
            return myAns
        #case -II
        if root.val!=left.univalue and root.val!=right.univalue:
            ans[0] = max(ans[0],1)
            myAns.length = 1
            return myAns
        #case -III
        if root.val==left.univalue:
            ans[0] = max(ans[0],left.length+1)
            myAns.length = left.length+1
            return myAns
        #case- IV
```

```
        if root.val==right.univalue:
            ans[0] = max(ans[0],right.length+1)
            myAns.length = right.length+1
            return myAns
```