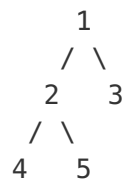
Diameter of Binary Tree

Given a binary tree, you need to compute the length of the diameter of the tree. The diameter of a binary tree is the length of the **longest** path between any two nodes in a tree. This path may or may not pass through the root.

Example:
Given a binary tree



Return **3**, which is the length of the path [4,2,1,3] or [5,2,1,3].

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

Solution 1

For `every` node, length of longest path which `pass it` = MaxDepth of its left subtree + MaxDepth of its right subtree.

```
public class Solution {
    int max = 0;

    public int diameterOfBinaryTree(TreeNode root) {
        maxDepth(root);
        return max;
    }

    private int maxDepth(TreeNode root) {
        if (root == null) return 0;

        int left = maxDepth(root.left);
        int right = maxDepth(root.right);

        max = Math.max(max, left + right);

        return Math.max(left, right) + 1;
    }
}
```

written by shawngao original link here

Solution 2

Let's calculate the depth of a node in the usual way: max(depth of node.left, depth of node.right) + 1. While we do, a path "through" this node uses 1 + (depth of node.left) + (depth of node.right) nodes. Let's search each node and remember the highest number of nodes used in some path. The desired length is 1 minus this number.

```
def diameterOfBinaryTree(self, root):
    self.best = 1
    def depth(root):
        if not root: return 0
        ansL = depth(root.left)
        ansR = depth(root.right)
        self.best = max(self.best, ansL + ansR + 1)
        return 1 + max(ansL, ansR)

    depth(root)
    return self.best - 1
```

written by awice original link here

Solution 3

```
class Solution {
public:
    int maxdiadepth = 0;

    int dfs(TreeNode* root){
        if(root == NULL) return 0;

        int leftdepth = dfs(root->left);
        int rightdepth = dfs(root->right);

        if(leftdepth + rightdepth > maxdiadepth) maxdiadepth = leftdepth + rightd
epth;
        return max(leftdepth +1, rightdepth + 1);
    }

    int diameterOfBinaryTree(TreeNode* root) {
        dfs(root);

        return maxdiadepth;
    }
};
```

written by Sublele original link here