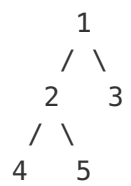
Diameter of Binary Tree

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

```
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

Solution 3

```
public class Solution {
    public int diameterOfBinaryTree(TreeNode root) {
        if(root == null){
            return 0;
        }
       int dia = depth(root.left) + depth(root.right);
       int ldia = diameterOfBinaryTree(root.left);
       int rdia = diameterOfBinaryTree(root.right);
       return Math.max(dia,Math.max(ldia,rdia));

    }
    public int depth(TreeNode root){
        if(root == null){
            return 0;
        }
        return 1+Math.max(depth(root.left), depth(root.right));
    }

}
```

written by sarojini original link here