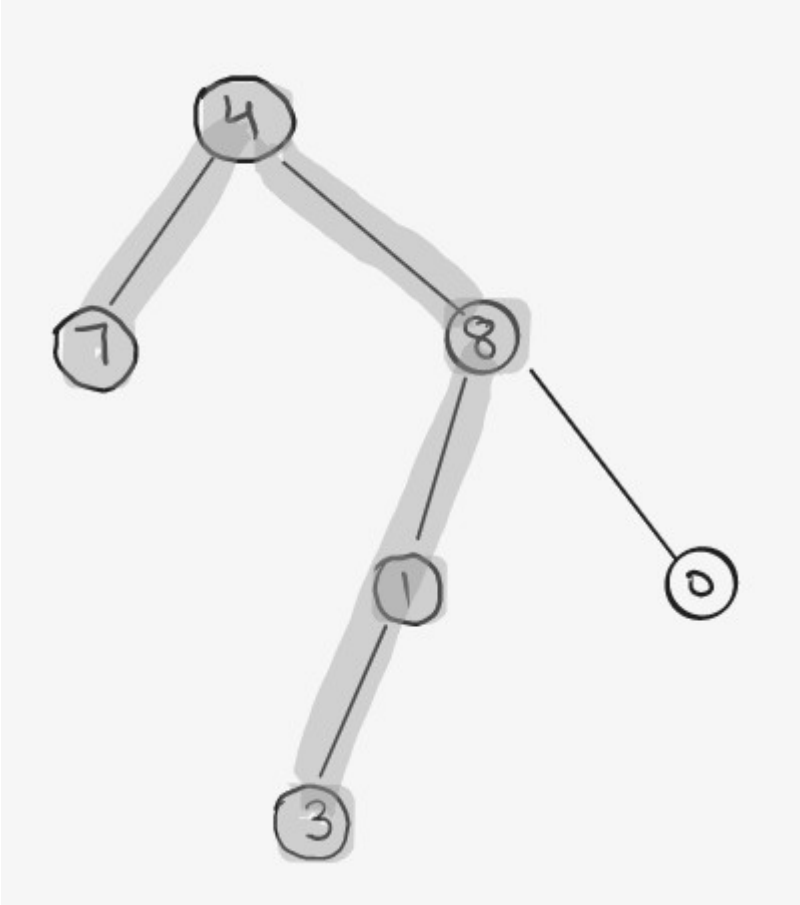
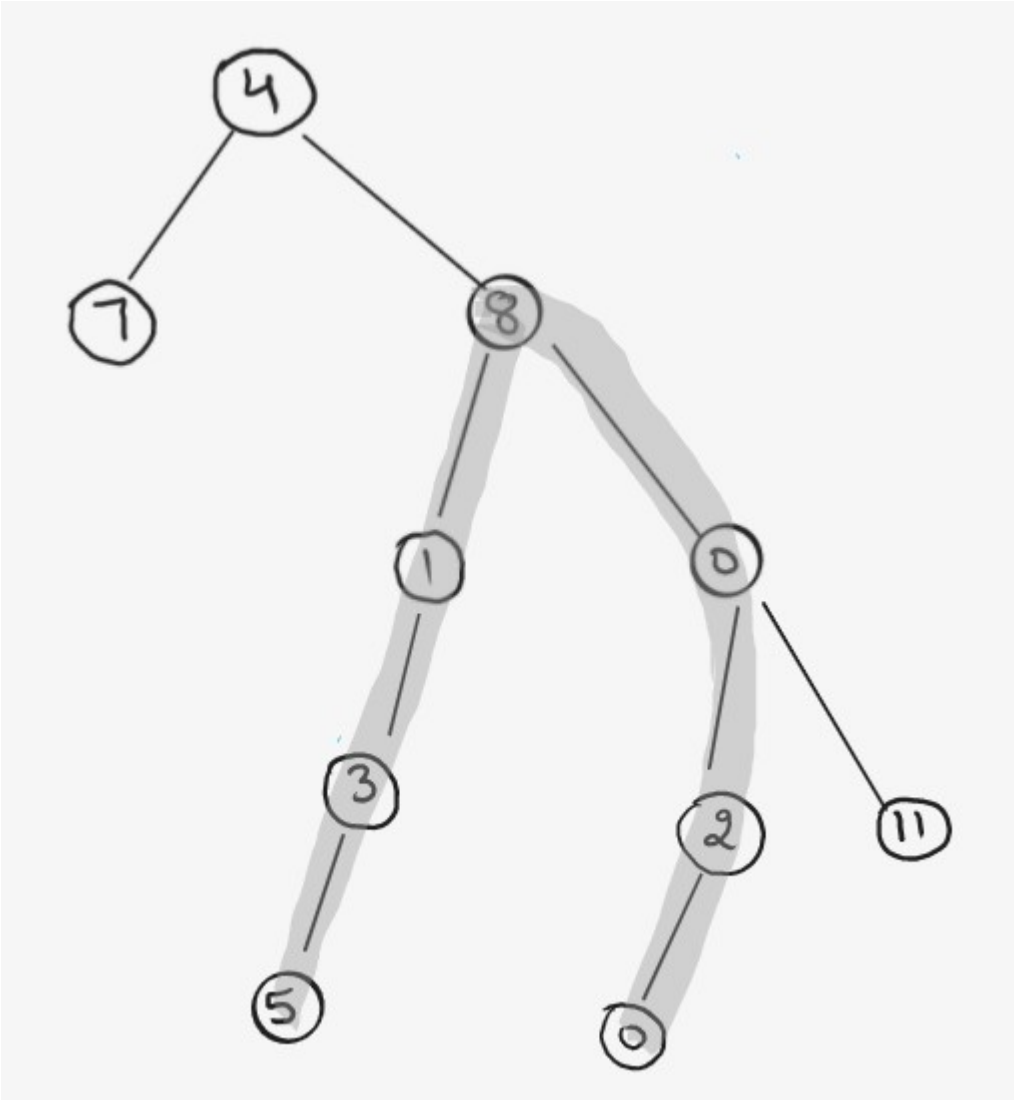
Calculate the Diameter of a Binary Tree
Problem Statement: Find the Diameter of a Binary Tree. **Diameter** is the length of the longest path between any 2 nodes in the tree and this path may or may not pass from the root.
Example 1:
Input Format: Given the root of Binary Tree



Result: 4
Explanation: Longest Path available is 7 – 4 – 8 – 1 – 3 of length 4
Example 2:
Input Format: Given the root of Binary Tree



Result: 6
Explanation: Longest Path available is 5 – 3 – 1 – 8 – 0 – 2 – 0 of length 6. (Path is not Passing from root).

Solutions for Diameter of a Binary Tree

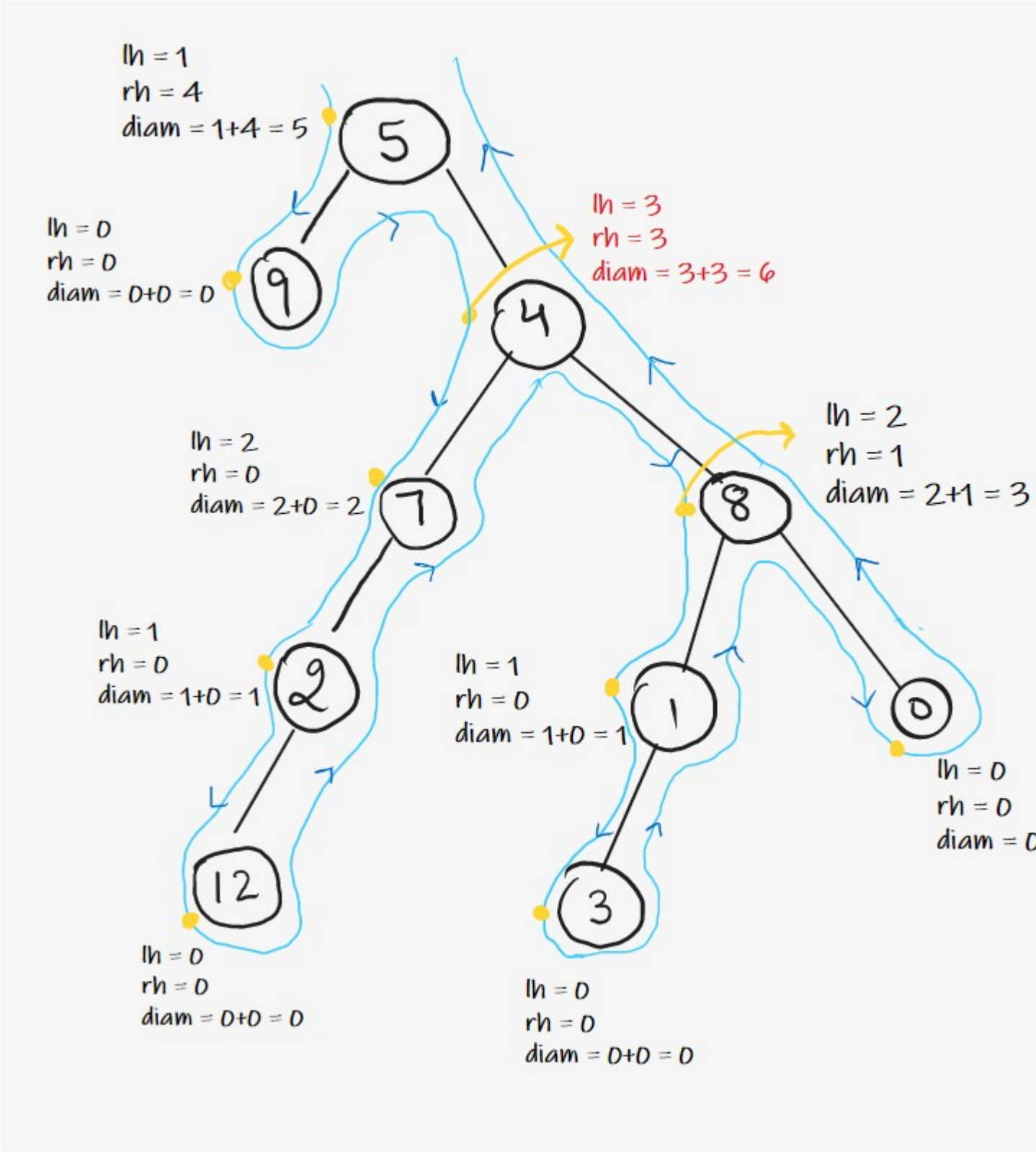
Disclaimer: *Don't jump directly to the solution, try it out yourself first.*
Solution 1: Naive approach
Intuition :
The idea is to consider every node as a *Curving Point* in diameter. For our understanding, we can define the curving point as the node on the diameter path which has the maximum height. In the above examples, the Curving Point is node **4** in Example 1 and node **8** in Example 2.
Now, if we observe carefully, we can see that diameter of the tree can be defined as left subtree height + right subtree height from the Curving Point.
So, the idea to find the Curving Point is, consider every node in the tree as a curving point and calculate the diameter for every curving point and return the maximum of all diameters.

Diameter at given Curving Point = Left Height + Right Height

Approach :

- Traverse the tree recursively.
- At every node, calculate height of left and right subtrees.
- Calculate the diameter for every node using the above formula.
- Calculate the maximum of all diameters. This can be done simply using a variable passed by reference in the recursive calls or a global static variable.

dry-run :



Start traversing the tree.

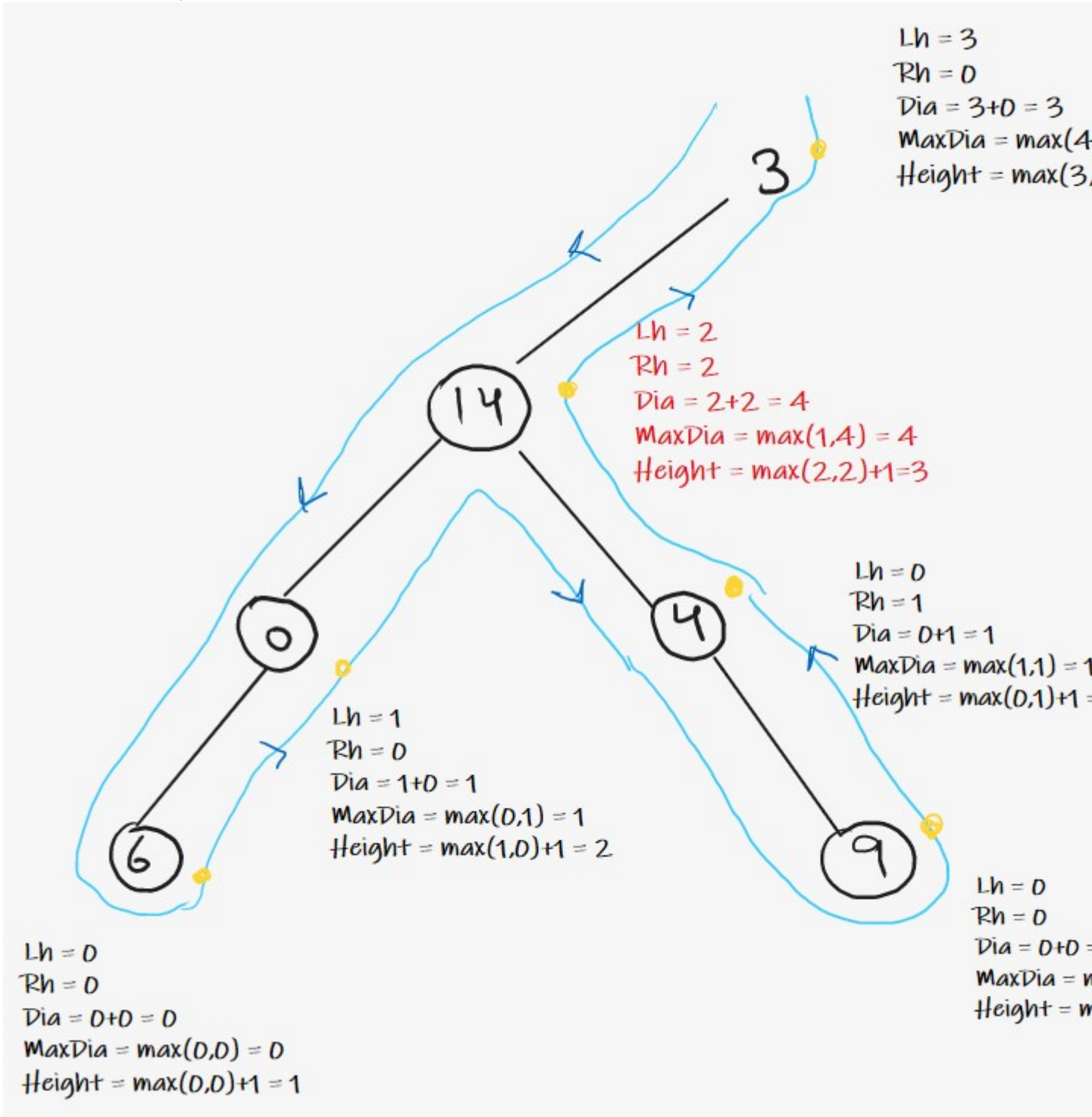
- Initialize Maximum Diameter variable with Integer.MIN_VALUE.
- Reach on **Node 5** , call Height Function , Left height = 1 , Right height = 4 so Diameter is (1 + 4) = 5. Hence , Maximum Diameter = Max(Integer.MIN_VALUE , 5) = 5.
- Reach on **Node 9** , call Height Function , Left height = 0 , Right height = 0 so Diameter is (0 + 0) = 0. Hence , Maximum Diameter = Max(5 , 0) = 5.
- Reach on **Node 4** , call Height Function , Left height = 3 , Right height = 3 so Diameter is (3 + 3) = 6. Hence , Maximum Diameter = Max(5 , 6) = 6.
- Reach on **Node 7** , call Height Function , Left height = 2 , Right height = 0 so Diameter is (2 + 0) = 2. Hence , Maximum Diameter = Max(6,2) = 6.
- Reach on **Node 2** , call Height Function , Left height = 1 , Right height = 0 so Diameter is (1 + 0) = 1. Hence , Maximum Diameter = Max(6 , 1) = 6.
- Reach on **Node 12** , call Height Function , Left height = 0 , Right height = 0 so Diameter is (0 + 0) = 0. Hence , Maximum Diameter = Max(6 , 0) = 6.
- Reach on **Node 8** , call Height Function , Left height = 2 , Right height = 1 so Diameter is (2 + 1) = 3. Hence , Maximum Diameter = Max(6 , 3) = 6.
- Reach on **Node 1** , call Height Function , Left height = 1 , Right height = 0 so Diameter is (1 + 0) = 1. Hence , Maximum Diameter = Max(6 , 1) = 6.

- Reach on **Node 3** , call Height Function , Left height = 0 , Right height = 0 so Diameter is (0 + 0) = 0. Hence , Maximum Diameter = Max(6 , 0) = 6.
- Reach on **Node 0** , call Height Function , Left height = 0 , Right height = 0 so Diameter is (0 + 0) = 0. Hence , Maximum Diameter = Max(6 , 0) = 6.

Time Complexity: O(N*N) (For every node, Height Function is called which takes O(N) time hence for every node it becomes N*N)
Space Complexity: O(1) (Extra Space) **+ O(H)** (Recursive Stack Space where "**H**" is the height of tree)
Solution 2: Post Order Traversal
Intuition :
Is it possible to optimize the above solution further? Which operation do you think is very repetitive in nature in the above solution?
Height of the subtrees.
Can we use postorder traversal to calculate everything in a single traversal of the tree?
Yes, as in post-order traversal, we have to completely traverse the left and right subtree before visiting the root node.
So, the idea is to use post-order traversal and keep calculating the height of the left and right subtrees. Once we have the heights at the current node, we can easily calculate both the diameter and height of the current node.
Approach :

- Start traversing the tree recursively and do work in Post Order.
- In the Post Order of every node , calculate diameter and height of the current node.
- If current diameter is maximum then update the variable used to store the maximum diameter.
- Return height of current node to the previous recursive call.

Dry Run :
In Post Order, Start traversing the tree:



- Reach on **Node 6 ,** Left height = 0 as left == null , Right height = 0 as right == null so Diameter is (0 + 0) = 0. Hence , Maximum Diameter = Max(0 , 0) = 0 and return height = max(0,0)+1 = 1.
- Reach on **Node 0**, Left height = 1 , Right height = 0 as right == null so Diameter is (1 + 0) = 1. Hence , Maximum Diameter = Max(0 , 1) = 1 and return height = max(1,0)+1 = 2.
- Reach on **Node 9 ,** Left height = 0 as left == null , Right height = 0 as right == null so Diameter is (0 + 0) = 0. Hence , Maximum Diameter = Max(1 , 0) = 1 and return height = max(0,0)+1 = 1.
- Reach on **Node 4 ,** Left height = 0 as left == null , Right height = 1 , so Diameter is (0 + 1) = 1. Hence , Maximum Diameter = Max(1 , 1) = 1 and return height = max(0,1)+1 = 2.
- Reach on **Node 14 ,** Left height = 2 , Right height = 2 , so Diameter is (2 + 2) = 4. Hence , Maximum Diameter = Max(1 , 4) = 4 and return height = max(2,2)+1 = 3.
- Reach on **Node 3 ,** Left height = 3 , Right height = 0 as right == null , so Diameter is (3 + 0) = 3. Hence , Maximum Diameter = Max(4 , 3) = 4 and return height = max(3,0)+1 = 4.
- Hence , the maximum diameter is 4 .

Code:

- C++ Code
- Java Code

```
class Solution {
public:
    int diameterOfBinaryTree(TreeNode* root) {
        int diameter = 0;
        height(root, diameter);
        return diameter;
    }

private:
    int height(TreeNode* node, int& diameter) {

        if (!node) {
            return 0;
        }

        int lh = height(node->left, diameter);
        int rh = height(node->right, diameter);

        diameter = max(diameter, lh + rh);

        return 1 + max(lh, rh);
    }
};
```

Time Complexity: O(N)
Space Complexity: O(1) Extra Space + O(H) Recursion Stack space (Where “**H**” is the height of binary tree)