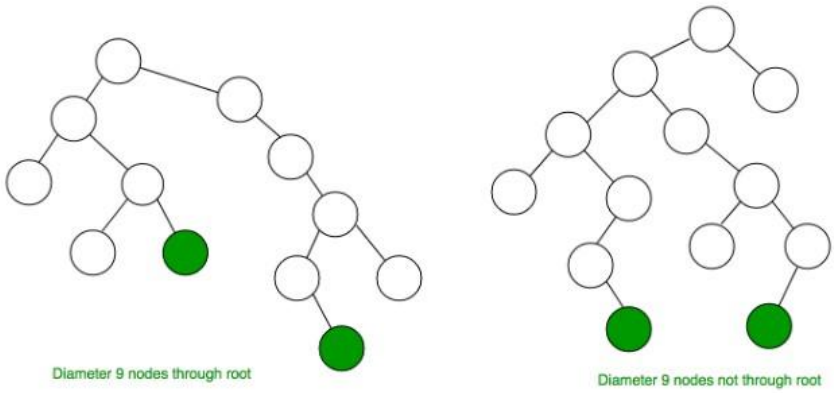
Diameter of a Binary Tree

The **diameter** of a tree (sometimes called the width) is the number of nodes on the longest path between two end nodes. The diagram below shows two trees each with diameter nine, the leaves that form the ends of a longest path are shaded (note that there is more than one path in each tree of length nine, but no path longer than nine nodes).



Solution: The diameter of a tree T is the largest of the following quantities:

- The diameter of T's left subtree.
- The diameter of T's right subtree.
- The longest path between leaves that goes through the root of T (this can be computed from the heights of the subtrees of T).

The longest path between leaves that goes through a particular node say, ***nd*** can be calculated as:

> *1 + height of left subtree of nd + height of right subtree of nd*

Therefore, final **Diameter** of a node can be calculated as:

> *Diameter = maximum(lDiameter, rDiameter, 1 + lHeight + rHeight)*
> Where,
> **lDiameter** = Diameter of left subtree
> **rDiameter** = Diameter of right subtree
> **lHeight** = Height of left subtree
> **rHeight** = Height of right subtree

Implementation:

C++ Java

```cpp
// C++ program to calculate Diameter of a Binary Tree
#include <bits/stdc++.h>
using namespace std;
// Binary Tree Node
struct node
{
    int data;
    struct node* left, *right;
};
// Function to create a new node of tree and returns pointer
struct node* newNode(int data);
// Function to Compute height of a tree
int height(struct node* node);
// Function to get diameter of a binary tree
int diameter(struct node * tree)
{
    /* base case where tree is empty */
    if (tree == NULL)
        return 0;
    /* get the height of left and right sub-trees */
    int lheight = height(tree->left);
    int rheight = height(tree->right);
    /* get the diameter of left and right sub-trees */
    int ldiameter = diameter(tree->left);
    int rdiameter = diameter(tree->right);
    /* Return max of following three
    1) Diameter of left subtree
    2) Diameter of right subtree
    3) Height of left subtree + height of right subtree + 1 */
    return max(lheight + rheight + 1, max(ldiameter, rdiameter));
}
/* UTILITY FUNCTIONS TO TEST diameter() FUNCTION */
/* The function Compute the "height" of a tree. Height is the
    number f nodes along the longest path from the root node
    down to the farthest leaf node.*/
int height(struct node* node)
{
    /* base case tree is empty */
    if(node == NULL)
        return 0;
    /* If tree is not empty then height = 1 + max of left height and right heights */
    return 1 + max(height(node->left), height(node->right));
}
/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
struct node* newNode(int data)
{
    struct node* node = (struct node*) malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;
    return(node);
}
// Driver Code
int main()
```

```
{

    /* Constructed binary tree is
             1
            / \
           2   3
          / \
         4     5
    */
    struct node *root = newNode(1);
    root->left      = newNode(2);
    root->right     = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);
    cout<<"Diameter of the given binary tree is "<<diameter(root);
    return 0;
}
```

Output:

```
Diameter of the given binary tree is 4
```