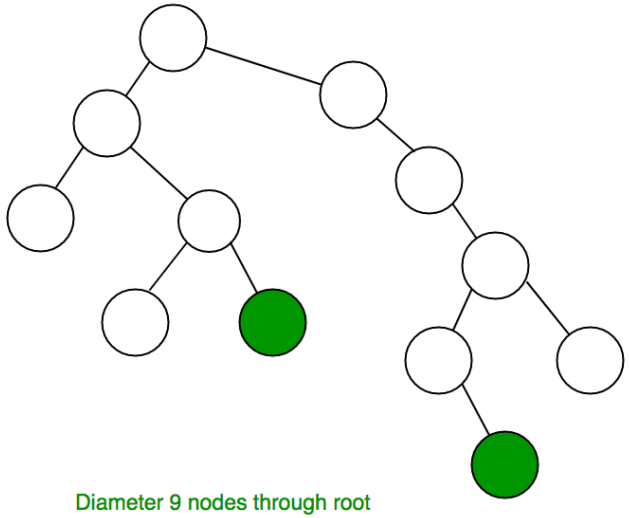
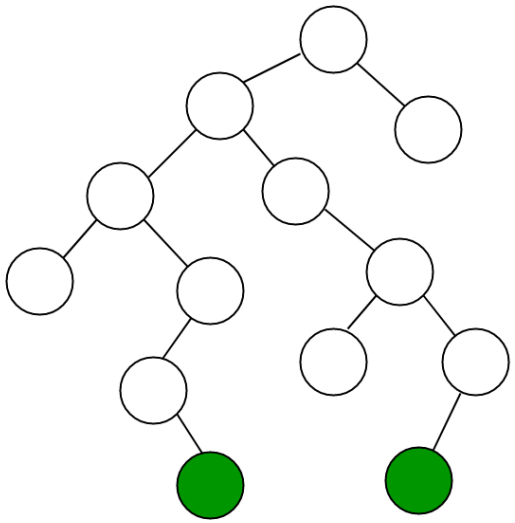
Dash
All
Articles
Videos
Problems
Quiz
Prev
Next

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).



Diameter 9 nodes through root



Diameter 9 nodes not through root

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

```
// Recursive optimized Java program to find the diameter of a
// Binary Tree

/* Class containing left and right child of current
 node and key value*/
class Node
{
    int data;
    Node left, right;

    public Node(int item)
    {
        data = item;
        left = right = null;
    }
}

/* Class to print the Diameter */
class BinaryTree
{
    Node root;

    /* Method to calculate the diameter and return it to main */
    int diameter(Node root)
    {
        /* base case if tree is empty */
        if (root == null)
            return 0;

        /* get the height of left and right sub trees */
        int lheight = height(root.left);
        int rheight = height(root.right);
```

```
        /* get the diameter of left and right subtrees */
        int ldiameter = diameter(root.left);
        int rdiameter = diameter(root.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 Math.max(lheight + rheight + 1,
                        Math.max(ldiameter, rdiameter));

    }

    /* A wrapper over diameter(Node root) */
    int diameter()
    {
        return diameter(root);
    }

    /*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.*/
    static int height(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 + Math.max(height(node.left), height(node.right)));
    }

    public static void main(String args[])
    {
        /* creating a binary tree and entering the nodes */
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
```

```
        tree.root.left.right = new Node(5);

        System.out.println("The diameter of given binary tree is : "
                           + tree.diameter());
    }
}
```


Output:

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

Time Complexity: O(N^2), where N is the number of nodes in the binary tree.

Auxiliary Space: O(N) for call stack

Efficient Approach: To solve the problem follow the below idea:

> The above implementation can be optimized by calculating the height in the same recursion rather than calling a height() separately.

Below is the implementation of the above approach:

C++ | Java | Python3 | C# | Javascript

```
// Recursive Java program to find the diameter of a
// Binary Tree

// Class containing left and right child of current
```

```
// node and key value
class Node {
    int data;
    Node left, right;

    public Node(int item)
    {
        data = item;
        left = right = null;
    }
}

// A utility class to pass height object
class Height {
    int h;
}

// Class to print the Diameter
class BinaryTree {
    Node root;

    // define height =0 globally and call
    // diameterOpt(root,height) from main
    int diameterOpt(Node root, Height height)
    {
        // lh --> Height of left subtree
        // rh --> Height of right subtree
        Height lh = new Height(), rh = new Height();

        if (root == null) {
            height.h = 0;
            return 0; // diameter is also 0
        }
        /*
                ldiameter --> diameter of left subtree
                rdiameter --> Diameter of right subtree
                Get the heights of left and right subtrees
            in lh and rh. And store the returned values in
            ldiameter and ldiameter*/
            int ldiameter = diameterOpt(root.left, lh);
            int rdiameter = diameterOpt(root.right, rh);
```

```
        // Height of current node is max of heights of left
        // and right subtrees plus 1
        height.h = Math.max(lh.h, rh.h) + 1;

        return Math.max(lh.h + rh.h + 1,
                        Math.max(ldiameter, rdiameter));
    }

    // A wrapper over diameter(Node root)
    int diameter()
    {
        Height height = new Height();
        return diameterOpt(root, height);
    }

    // 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.
    static int height(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
```



```
    }

    // Driver Code
    public static void main(String args[])
    {
        // creating a binary tree and entering the nodes
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
```

```
            tree.root.left.left = new Node(4);
            tree.root.left.right = new Node(5);

            // Function Call
            System.out.println(
                "The diameter of given binary tree is : "
                + tree.diameter());
        }
    }
```

Output

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

Time Complexity: O(N)
Auxiliary Space: O(N) due to recursive calls.

Dash
All
Articles
Videos
Problems
Quiz