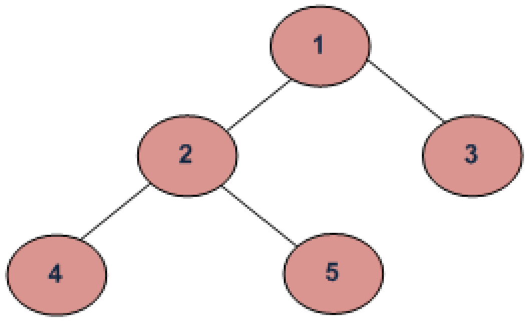
Dash
All
Articles
Videos
Problems
Quiz
« Prev
Next »

Height of Binary Tree

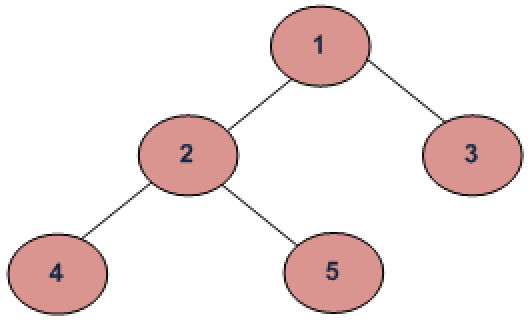
Given a binary tree, the task is to find the height of the tree. Height of the tree is the number of edges in the tree from the root to the deepest node, Height of the empty tree is **0**.



Recursively calculate height of **left** and **right** subtrees of a node and assign height to the node as **max of the heights of two children plus 1**. See below pseudo code and program for details.

Illustration:

> *Consider the following graph:*



maxDepth('1') = max(maxDepth('2'), maxDepth('3')) + 1 = 2 + 1

because recursively
maxDepth('2') = max (maxDepth('4'), maxDepth('5')) + 1 = 1 + 1 and (as height of both '4' and '5' are 1)
maxDepth('3') = 1

Follow the below steps to Implement the idea:

- Recursively do a Depth-first search.
- If the tree is empty then return -1
- Otherwise, do the following
  - Get the max depth of the left subtree recursively i.e. call maxDepth(tree->left-subtree)
  - Get the max depth of the right subtree recursively i.e. call maxDepth(tree->right-subtree)
  - Get the max of max depths of **left** and **right** subtrees and **add 1** to it for the current node.
    - $max_depth = max(maxdeptofleftsubtree, maxdepthofrightsubtree) + 1$
- Return max_depth.

Below is the Implementation of the above approach:

C++ Java

```java
 // Java program to find height of tree

// A binary tree node
class Node {
    int data;
    Node left, right;

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

class BinaryTree {
    Node root;

    /* Compute the "maxDepth" of a tree -- the number of
    nodes along the longest path from the root node
    down to the farthest leaf node.*/
    int maxDepth(Node node)
    {
        if (node == null)
            return 0;
        else {
            /* compute the depth of each subtree */
            int lDepth = maxDepth(node.left);
            int rDepth = maxDepth(node.right);
            /* use the larger one */
            if (lDepth > rDepth)
                return (lDepth + 1);
```

```
        }
    }

    /* Driver program to test above functions */
    public static void main(String[] args)
    {
        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("Height of tree is "
                        + tree.maxDepth(tree.root));
    }
}
```

All
Articles
Videos
Problems
Quiz

Output

Dash
All
Articles
Videos
Problems
Quiz

```
Height of tree is 3
```

Time Complexity: O(N) (Please see our post Tree Traversal for details)
Auxiliary Space: O(N) due to recursive stack.