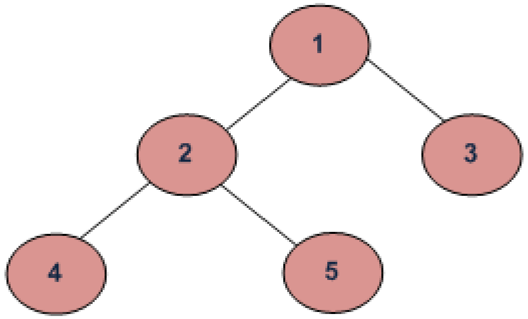
Dash
All
Articles
Videos
Problems
Quiz
« Prev
Next »

Size of Binary Tree

Size of a tree is the number of elements present in the tree. Size of the below tree is 5.



Size() function recursively calculates the size of a tree. It works as follows:
Size of a tree = Size of left subtree + 1 + Size of right subtree.

Algorithm:

```
size(tree)
1. If tree is empty then return 0
2. Else
     (a) Get the size of left subtree recursively  i.e., call
          size( tree->left-subtree)
     (a) Get the size of right subtree recursively  i.e., call
```

```
          size( tree->right-subtree)
    (c) Calculate size of the tree as following:
            tree_size  =  size(left-subtree) + size(right-
                               subtree) + 1
    (d) Return tree_size
```

Dash
All
Articles
Videos
Problems
Quiz

C++ Java

```java
 // A recursive Java program to calculate the size of the
// 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;
    }
```

Courses Tutorials Jobs Practice Contests

P

«
»

```java
/* Class to find size of Binary Tree */
class BinaryTree {
    Node root;
```

Dash
All
Articles
Videos
Problems
Quiz
«
»

```
    // Function to return the size of binary tree
    int size() { return size(root); }

    /* computes number of nodes in tree */
    int size(Node node)
    {
        if (node == null)
            return 0;
        else
            return (size(node.left) + 1 + size(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 size of binary tree is : "
                           + tree.size());
    }
}
```

Output:

```
Size of the tree is 5
```

Dash
All
Articles
Videos
Problems
Quiz
«
»

Time Complexity: O(N)

As every node is visited once.

Auxiliary Space: O(N)

The extra space is due to the recursion call stack and the worst case occurs when the tree is either left skewed or right skewed.

Since this program is similar to traversal of tree, time and space complexities will be same as Tree traversal

Mark as Read

Report An Issue

If you are facing any issue on this page. Please let us know.