Print Right View of a Binary Tree

Given a Binary Tree, print Right view of it. Right view of a Binary Tree is set of nodes visible when tree is visited from Right side.

```
Right view of following tree is 1 3 7 8

          1
       /     \
     2        3
   /   \     /  \
  4     5   6    7
                  \
                   8
```

We strongly recommend to minimize the browser and try this yourself first.

The Right view contains all nodes that are last nodes in their levels. A simple solution is to do level order traversal and print the last node in every level.

The problem can also be solved using simple recursive traversal. We can keep track of level of a node by passing a parameter to all recursive calls. The idea is to keep track of maximum level also. And traverse the tree in a manner that right subtree is visited before left subtree. Whenever we see a node whose level is more than maximum level so far, we print the node because this is the last node in its level (Note that we traverse the right subtree before left subtree). Following is C implementation of this approach.

C

```
// C program to print right view of Binary Tree
#include<stdio.h>
#include<stdlib.h>

struct Node
{
    int data;
    struct Node *left, *right;
};
```

```
// A utility function to create a new Binary Tree Node
struct Node *newNode(int item)
{
    struct Node *temp =  (struct Node *)malloc(sizeof(struct Node));
    temp->data = item;
    temp->left = temp->right = NULL;
    return temp;
}

// Recursive function to print right view of a binary tree.
void rightViewUtil(struct Node *root, int level, int *max_level)
{
    // Base Case
    if (root==NULL)  return;

    // If this is the last Node of its level
    if (*max_level < level)
    {
        printf("%d\t", root->data);
        *max_level = level;
    }

    // Recur for right subtree first, then left subtree
    rightViewUtil(root->right, level+1, max_level);
    rightViewUtil(root->left, level+1, max_level);
}

// A wrapper over rightViewUtil()
void rightView(struct Node *root)
{
    int max_level = 0;
    rightViewUtil(root, 1, &max_level);
}

// Driver Program to test above functions
int main()
{
    struct Node *root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);
    root->right->left = newNode(6);
    root->right->right = newNode(7);
    root->right->left->right = newNode(8);

    rightView(root);

    return 0;
}
```

Java

```
// Java program to print right view of binary tree

// A binary tree node
class Node {

    int data;
    Node left, right;

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

// class to access maximum level by reference
class Max_level {

    int max_level;
}

class BinaryTree {

    Node root;
    Max_level max = new Max_level();

    // Recursive function to print right view of a binary tree.
    void rightViewUtil(Node node, int level, Max_level max_level) {

        // Base Case
        if (node == null)
            return;

        // If this is the last Node of its level
        if (max_level.max_level < level) {
            System.out.print(node.data + " ");
            max_level.max_level = level;
        }

        // Recur for right subtree first, then left subtree
        rightViewUtil(node.right, level + 1, max_level);
        rightViewUtil(node.left, level + 1, max_level);
    }

    void rightView()
    {
        rightView(root);
    }

    // A wrapper over rightViewUtil()
    void rightView(Node node) {

        rightViewUtil(node, 1, max);
    }

    // Driver program to test the 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);
        tree.root.right.left = new Node(6);
        tree.root.right.right = new Node(7);
        tree.root.right.left.right = new Node(8);

        tree.rightView();

        }
}

// This code has been contributed by Mayank Jaiswal
```

Python

```
# Python program to print right view of Binary Tree

# A binary tree node
class Node:
    # A constructor to create a new Binary tree Node
    def __init__(self, item):
        self.data = item
        self.left = None
        self.right = None

# Recursive function to print right view of Binary Tree
# used max_level as reference list ..only max_level[0]
# is helpful to us
def rightViewUtil(root, level, max_level):

    # Base Case
    if root is None:
        return

    # If this is the last node of its level
    if (max_level[0] < level):
        print "%d   " %(root.data),
        max_level[0] = level

    # Recur for right subtree first, then left subtree
    rightViewUtil(root.right, level+1, max_level)
    rightViewUtil(root.left, level+1, max_level)

def rightView(root):
    max_level = [0]
    rightViewUtil(root, 1, max_level)


# Driver program to test above function
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)
root.right.left = Node(6)
root.right.right = Node(7)
root.right.left.right = Node(8)

rightView(root)

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

Output:

```
1      3      7     8
```

Time Complexity: The function does a simple traversal of the tree, so the complexity is O(n).

This article is contributed by **Shalki Agarwal**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

74 Comments Category: Trees

Related Posts:

- Check if removing an edge can divide a Binary Tree in two halves
- Check sum of Covered and Uncovered nodes of Binary Tree
- Lowest Common Ancestor in a Binary Tree | Set 2 (Using Parent Pointer)
- Construct a Binary Search Tree from given postorder
- BFS vs DFS for Binary Tree
- Maximum difference between node and its ancestor in Binary Tree
- Inorder Non-threaded Binary Tree Traversal without Recursion or Stack
- Check if leaf traversal of two Binary Trees is same?

Like Share 12 people like this.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.