GeeksforGeeks
A computer science portal for geeks

GeeksQuiz

Login

Home Algorithms DS GATE Interview Corner Q&A C C++ Java Books Contribute Ask a Q About

Array Bit Magic C/C++ Articles GFacts Linked List MCQ Misc Output String Tree Graph

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.

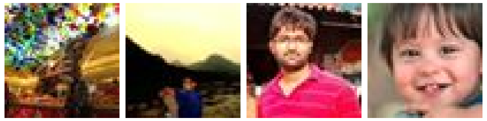
```
Right view of following tree is 1 3 7 8

          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 first 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 first node in its level (Note that we traverse the right subtree before left subtree). Following is C implementation of this approach.

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



Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion

```
#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 first 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);
```

Geometric Algorithms

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

```
    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;
}
```

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

Related Tpoics:

- Print a Binary Tree in Vertical Order | Set 2 (Hashmap based Method)
- Red-Black Tree | Set 3 (Delete)
- Construct a tree from Inorder and Level order traversals
- Print all nodes at distance k from a given node
- Print a Binary Tree in Vertical Order | Set 1
- Interval Tree
- Check if a given Binary Tree is height balanced like a Red-Black Tree
- Print all nodes that are at distance k from a leaf node

9 Tweet 3 1

Writing code in comment? Please use **ideone.com** and share the link here.

16 Comments **GeeksforGeeks**

Sort by Newest ▾

Join the discussion…

Jaks C · 2 days ago

The simplest solution is,

```
private void printRightView(Node node){

if (node == null) return;

System.out.print("->" + node.getKey());

printRightView(node.getRight());

}
```

695 Subscribe

Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...

Backtracking | Set 7 (Sudoku) · 25 minutes ago

RVM Can someone please elaborate this Qs from above...

Flipkart Interview | Set 6 · 45 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...

Software Engineering Lab, Samsung Interview | Set 2 · 45 minutes ago

@meya Working solution for question 2 of 4f2f round....

Amazon Interview | Set 53 (For SDE-1) · 1 hour ago

sandeep void rearrange(struct node *head) {...

Given a linked list, reverse alternate nodes and append at the end · 2 hours ago

Neha I think that is what it should return as, in...

Find depth of the deepest odd level leaf node · 2 hours ago

}

Complexity is O(log n)

Reply • Share ›

Naveen Bansal • 18 days ago

```
//make a global array view
// maintain a variable max_height to store the height of tree
//now just do inorder traversal of the tree and store the value of node in //view a
height of the tree

//finally print the view array from index 0 to max_height

int view[20];

int max_height;

void inorder(struct Node *root, int height)
{
if(height > max_height)
max_height= height;

if(root==NULL)
return;

inorder(root->left,height+1);
view[height]=root->data;
inorder(root->right,height+1);
}
```

Reply • Share ›

Dexter • 23 days ago

```
void right(struct tree *root)

{
```

```
{

if(root==NULL)

return;

else

{

printf("%d\n",root->data);

if(root->right!=NULL)

right(root->right);

else

{
```

see more

^ | ⌄ • Reply • Share ›

Aliah ➦ Dexter • 21 days ago

This solution would not work for all cases. Suppose in the last level the
left subtree itself.

1 ^ | ⌄ • Reply • Share ›

Dexter ➦ Aliah • 18 days ago

thanks aliah for notifying me!!!
will correct it!!

^ | ⌄ • Reply • Share ›

swati • 24 days ago

Iterative code..

```
void print_right_view(struct node *root)

{

struct node *prev=NULL;

while(root!=NULL)

{

if(root->right)

{

printf("%d ",root->data);

prev=root;

root=root->right;
```

see more

Reply • Share ›

Guest • a month ago

http://effprog.wordpress.com/2...

1 | • Reply • Share ›

codex • a month ago

please tell me whether i can do this way....

```
#include<stdio.h>

#include<stdlib.h>

struct node{
```

```
int data;

struct node*left;

struct node*right;

};

struct node*newnode(int data)

{

struct node*temp=(struct node*)malloc(sizeof(struct node));
```

see more

^ | ⌄ • Reply • Share ›

RajKumar Rampalli ➦ codex • a month ago

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

(8 is the right child of 6) (6 is the left child of 3) (3 is the right child of ro

Your solution won't work for above tree, since the answer should be 1
recursion in your display code.

```
void display(struct node*root)
{
printf("\n%d",temp->data);
if (temp->right != NULL) display(temp->right);
else if (temp->left != NULL) display(temp->left);
}
```

// Initially call display with root as argument from main()

Suggestions are welcome.

⌃ | ⌄ · Reply · Share ›

RajKumar Rampalli ➦ RajKumar Rampalli · a month ago

The above display function also won;t work. We should conside
we remove 6 7 8 from the input tree. Because the output shoul
code then the output is 1 3 only which is wrong. So, Shalki Aga

⌃ | ⌄ · Reply · Share ›

bhaskar · a month ago

```
RightView( Struct Tree *t) {
if (t != null) {
print(t->data)
RightView(t->right);
}
}
```

Should be like this !!

⌃ | ⌄ · Reply · Share ›

Anshul Goel ➦ bhaskar · a month ago

This won't do the work. Check for the above case and remove nodes 6

1 ⌃ | ⌄ · Reply · Share ›

Jonathan ➦ Anshul Goel · a month ago

uh, don't get it. Remove 6,7,8 from the original tree?

So, returning 1,3 would be wrong then?

So, returning 1,3 would be wrong then?

Reply • Share ›

RajKumar Rampalli Jonathan • 25 days ago

Yes, it is wrong. The output should be 1 3 5 (right view c

Reply • Share ›

kaushik Lele • a month ago

We can achieve this instead of recursion. e.g. in Java we can write code this v

Node i = rootNode;

while(i != null){

System.out.println(i.getNodeVal());

i = i.getRightNode();

}

Reply • Share ›

Amol Ravande kaushik Lele • 24 days ago

void RightView(node* root)

{

while(root!=NULL)

{

cout<<root->value;

if(root->right==NULL)

{

```
root=root->left;
}
else
root=root->right;
}
}
```

⌃ | ⌄ • Reply • Share ›

Subscribe Add Disqus to your site