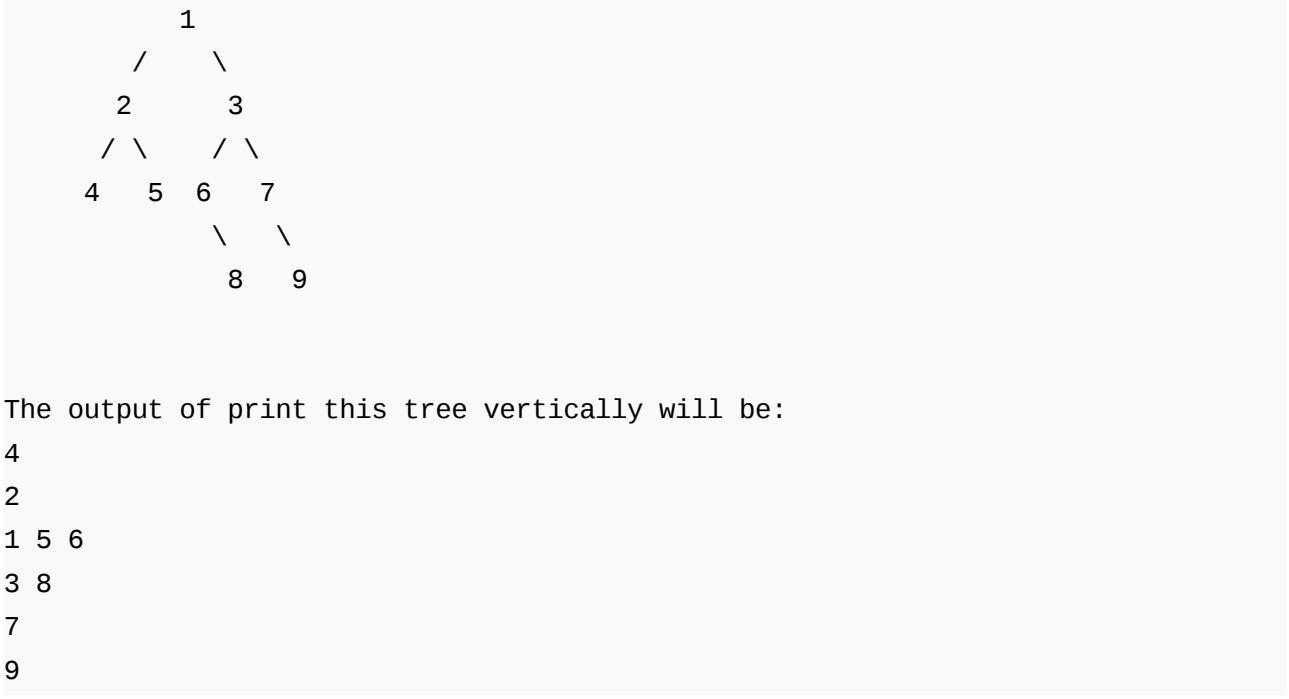
Print a Binary Tree in Vertical Order | Set 2 (Hashmap based Method)

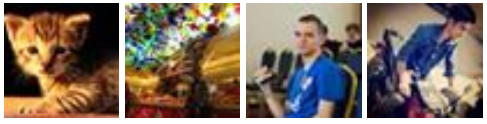
Given a binary tree, print it vertically. The following example illustrates vertical order traversal.




```
The output of print this tree vertically will be:
4
2
1 5 6
3 8
7
9
```

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

We have discussed a O(n^2) solution in the previous post. In this post, an efficient solution based on hash map is discussed. We need to check the Horizontal Distances from root for all nodes. If two nodes have the same Horizontal Distance (HD), then they are on same vertical line. The idea



of HD is simple. HD for root is 0, a right edge (edge connecting to right subtree) is considered as +1 horizontal distance and a left edge is considered as -1 horizontal distance. For example, in the above tree, HD for Node 4 is at -2, HD for Node 2 is -1, HD for 5 and 6 is 0 and HD for node 7 is +2.
We can do inorder traversal of the given Binary Tree. While traversing the tree, we can recursively calculate HDs. We initially pass the horizontal distance as 0 for root. For left subtree, we pass the Horizontal Distance as Horizontal distance of root minus 1. For right subtree, we pass the Horizontal Distance as Horizontal Distance of root plus 1. For every HD value, we maintain a list of nodes in a hasp map. Whenever we see a node in traversal, we go to the hash map entry and add the node to the hash map using HD as a key in map.

Following is C++ implementation of the above method. Thanks to Chirag for providing the below C++ implementation.

```
// C++ program for printing vertical order of a given binary tree
#include <iostream>
#include <vector>
#include <map>
using namespace std;

// Structure for a binary tree node
struct Node
{
    int key;
    Node *left, *right;
};

// A utility function to create a new node
struct Node* newNode(int key)
{
    struct Node* node = new Node;
    node->key = key;
    node->left = node->right = NULL;
    return node;
}

// Utility function to store vertical order in map 'm'
// 'hd' is horigontal distance of current node from root.
// 'hd' is initally passed as 0
void getVerticalOrder(Node* root, int hd, map<int, vector<int>> &m)
{
    // Base case
    if (root == NULL)
```


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

```
        return;

    // Store current node in map 'm'
    m[hd].push_back(root->key);

    // Store nodes in left subtree
    getVerticalOrder(root->left, hd-1, m);

    // Store nodes in right subtree
    getVerticalOrder(root->right, hd+1, m);
}

// The main function to print vertical oder of a binary tree
// with given root
void printVerticalOrder(Node* root)
{
    // Create a map and store vertical oder in map using
    // function getVerticalOrder()
    map < int,vector<int> > m;
    int hd = 0;
    getVerticalOrder(root, hd,m);

    // Traverse the map and print nodes at every horigontal
    // distance (hd)
    map< int,vector<int> > :: iterator it;
    for (it=m.begin(); it!=m.end(); it++)
    {
        for (int i=0; i<it->second.size(); ++i)
            cout << it->second[i] << " ";
        cout << endl;
    }
}

// Driver program to test above functions
int main()
{
    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);
    root->right->right->right = newNode(9);
    cout << "Vertical order traversal is \n";
    printVerticalOrder(root);
```

```
    return 0;
}
```

Output:

```
Vertical order traversal is
4
2
1 5 6
3 8
7
9
```

Time Complexity of hashing based solution can be considered as O(n) under the assumption that we have good hashing function that allows insertion and retrieval operations in O(1) time. In the above C++ implementation, map of STL is used. map in STL is typically implemented using a Self-Balancing Binary Search Tree where all operations take O(Logn) time. Therefore time complexity of above implementation is O(nLogn).

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

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

Related Tpoics:

- Print Right View of a Binary Tree
- 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

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

12 Comments GeeksforGeeks

Sort by Newest

prashnat · 3 days ago

set the x cordinate of all the nodes and nodes with same x value lie in the sam

```
#include<iostream>

#define min 999

using namespace std;
```

```
struct tnode

{

tnode* lchild;

int data;

tnode* rchild;

int x;

tnode(int d)
```

see more

• Reply • Share ›

dark_knight • 14 days ago

For this input , it is giving wrong answer ?

```
Node *root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->left->right = newNode(8);

root->left->left->right->right = newNode(9);

root->left->right = newNode(5);

root->right->left = newNode(6);
```

```
root->right->right = newNode(7);

root->right->left->right = newNode(10);
```

see more

Reply • Share ›

dark_knight dark_knight • 14 days ago

We should rather use either BFS or sort the vectors in the end

Reply • Share ›

amit isngh • 14 days ago

```
//date: 17th april 14
//by: amit kumar singh
#include<cstdio>
#include<iostream>
#include<cstdlib>
#include<queue>
using namespace std;
vector<int> v[100];
int min1=1000,max1=-1000;
struct node
{
int data;
struct node* left;
struct node* right;
};

typedef struct node * nodeptr;
queue<nodeptr> q;
```

see more

Reply • Share ›

Ankit Jain • 14 days ago

RECURSIVE SOLUTION

```
/*Print vertical tree*/
#include<stdio.h>
#include<stdlib.h>

int b[20];
struct BTree
{
int data;
struct BTree *left;
struct BTree *right;
};
int max(int a,int b)
{
if(a>b)
return a;
return b;
}
```

see more

  • Reply • Share ›

Ankit Jain • 14 days ago

Solution with QUEUE

```
/*Print vertical tree*/
#include<stdio.h>
#include<stdlib.h>

struct BTree
{
```

```
int data;
struct BTree *left;
struct BTree *right;
};

struct queue
{
struct BTree *b[100];
int front,rear;
}q;
```

see more

Reply • Share ›

Guest • 15 days ago

```
queue.enQueue(root);
queue.enQueue(null);
while(!queue.isEmpty){
temp = queue.deQueue();
if(temp==null)
queue.deQueue();
break line;
queue.enQueue(null);
}else {
print(temp.data);
if(temp.left!=null) queue.enQueue(temp.left);
if(temp.right!=null) queue.enQueue(temp.right);
}

}
```

Reply • Share ›

Test • 17 days ago

what if the node 5 has 2 children? The right child will get a positive number wh
paradigm.. I think the question is plain wrong and even if we consider it right, it
traversal.. Having said that, I am open for criticism..

1 ^ | v • Reply • Share ›

Guest • 20 days ago

Can you elaborate what do you mean by "vertical order" ?
Why 1,5,6 are treated to be in single column ?
Why 3,8 are treated to be in single column ?

^ | v • Reply • Share ›

Babu ➚ Guest • 18 days ago

Start from the root as horizontal distance as 0. Every time you move tc
you move to a right node do hd=hd+1. Now you need to print nodes ac
from root. For the tree given we have below hd
4 node as hd of -2
2 node as hd of -1
1 5 6 node as hd of 0
3 8 node as hd of +1
7 node as hd of +2
9 node as hd of +3

Hope it helps

^ | v • Reply • Share ›

Babu • 20 days ago

GFG you are doing a great job but what I see that we don't any collection of De
asked in many interviews. Can you post some Design Questions also and the

2 ^ | v • Reply • Share ›

Amber • 20 days ago

Amber • 20 days ago

This is same as finding columns in a tree and printing them out.

^ | v • Reply • Share ›

Subscribe Add Disqus to your site