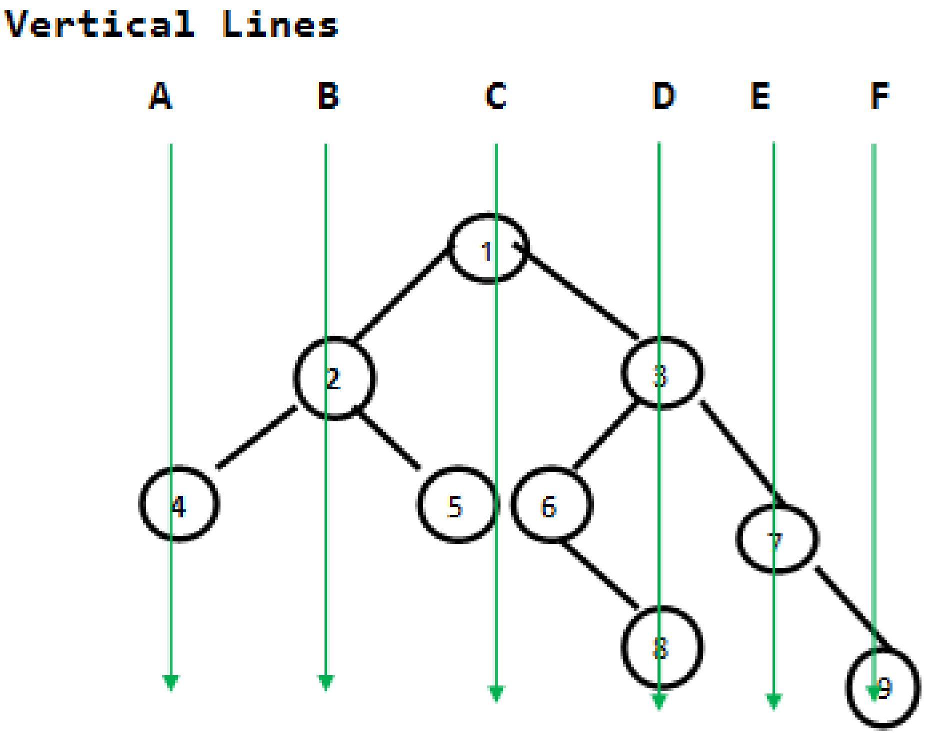
Print a Binary Tree in Vertical Order | Set 1

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

```
           1
        /    \
       2      3
      / \    / \
     4   5  6   7
             \   \
              8   9
```

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



Vertical order traversal is:

A- 4

B- 2

C- 1 5 6

D- 3 8

D- 3 8

E- 7

F- 9

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

The idea is to traverse the tree once and get the minimum and maximum horizontal distance with respect to root. For the tree shown above, minimum distance is -2 (for node with value 4) and maximum distance is 3 (For node with value 9).
Once we have maximum and minimum distances from root, we iterate for each vertical line at distance minimum to maximum from root, and for each vertical line traverse the tree and print the nodes which lie on that vertical line.

Algorithm:

```
// min --> Minimum horizontal distance from root
// max --> Maximum horizontal distance from root
// hd  --> Horizontal distance of current node from root
findMinMax(tree, min, max, hd)
     if tree is NULL then return;

     if hd is less than min then
           min = hd;
     else if hd is greater than max then
          *max = hd;

     findMinMax(tree->left, min, max, hd-1);
     findMinMax(tree->right, min, max, hd+1);
```

```
printVerticalLine(tree, line_no, hd)
     if tree is NULL then return;

     if hd is equal to line_no, then
           print(tree->data);
     printVerticalLine(tree->left, line_no, hd-1);
     printVerticalLine(tree->right, line_no, hd+1);
```

Implementation:

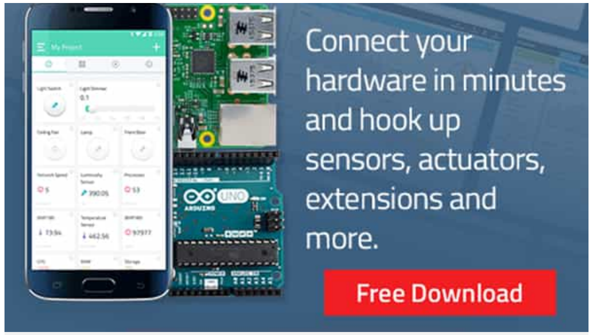
Following is the implementation of above algorithm.

C++ Java Python

```
#include <iostream>
using namespace std;

// A node of binary tree
struct Node
{
    int data;
    struct Node *left, *right;
};

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


Connect your hardware in minutes and hook up sensors, actuators, extensions and more.
Free Download

Popular Posts

- Top 10 Algorithms and Data Structures for Competitive Programming
- Top 10 algorithms in Interview Questions
- How to begin with Competitive Programming?
- Step by Step Guide for Placement Preparation
- Reflection in Java
- Memory Layout of C Programs
- Heavy Light Decomposition
- Sorted Linked List to Balanced BST
- Generics in Java
- Aho-Corasick Algorithm for Pattern Searching
- Insertion Sort , Binary Search , QuickSort , MergeSort , HeapSort

```
}

// A utility function to find min and max distances with
// to root.
void findMinMax(Node *node, int *min, int *max, int hd)
{
    // Base case
    if (node == NULL) return;

    // Update min and max
    if (hd < *min)  *min = hd;
    else if (hd > *max) *max = hd;

    // Recur for left and right subtrees
    findMinMax(node->left, min, max, hd-1);
    findMinMax(node->right, min, max, hd+1);
}

// A utility function to print all nodes on a given line_
// hd is horizontal distance of current node with respect
void printVerticalLine(Node *node, int line_no, int hd)
{
    // Base case
    if (node == NULL) return;

    // If this node is on the given line number
    if (hd == line_no)
        cout << node->data << " ";

    // Recur for left and right subtrees
    printVerticalLine(node->left, line_no, hd-1);
    printVerticalLine(node->right, line_no, hd+1);
}

// The main function that prints a given binary tree in
// vertical order
void verticalOrder(Node *root)
{
    // Find min and max distances with resepect to root
    int min = 0, max = 0;
```

- Common Interview Puzzles
- Interview Experiences
- Advanced Data Structures
- Design Patterns
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Geometric Algorithms
- Searching
- Sorting
- Analysis of Algorithms
- Mathematical Algorithms
- Randomized Algorithms
- Recursion
- Game Theory

Tags

Adobe Advanced Data Structure Amazon Arrays Articles C-Output C/C++ Puzzles cpp-library Divide and Conquer Dynamic Programming

```
    findMinMax(root, &min, &max, 0);

    // Iterate through all possible vertical lines starti
    // from the leftmost line and print nodes line by lir
    for (int line_no = min; line_no <= max; line_no++)
    {
        printVerticalLine(root, line_no, 0);
        cout << endl;
    }
}

// Driver program to test above functions
int main()
{
    // Create binary tree shown in above figure
    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" ;
    verticalOrder(root);

    return 0;
}
```

Run on IDE

Output:

```
Vertical order traversal is
```

Dynamic Programming Experienced Flipkart GATE GATE-CS-DS-&-Algo GBlog Graph Greedy Hash Internship Interview Experiences Java-Collections Linked Lists Mathematical Matrix MCQ Oracle Output Pattern Searching Project Python Python Searching Sorting Stack STL Strings subsequence Technical Scripter Trees XOR

```
4
2
1 5 6
3 8
7
9
```

Time Complexity: Time complexity of above algorithm is O(w*n) where w is width of Binary Tree and n is number of nodes in Binary Tree.
can be O(n) (consider a complete tree for example) and time complexity can become O(n^2).

This problem can be solved more efficiently using the technique discussed in this post. We will soon be discussing complete algorithm and implementation of more efficient method.

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

GATE CS Corner

Company Wise Coding Practice

Trees

Related Posts:

- Number of full binary trees such that each node is product of its children
- Print all nodes in a binary tree having K leaves
- Maximum sum of nodes in Binary tree such that no two are adjacent

- Find maximum level sum in Binary Tree
- Root to leaf paths having equal lengths in a Binary Tree
- Continuous Tree
- Find if there is a pair in root to a leaf path with sum equals to root’s data
- Check if there is a root to leaf path with given sequence

<< Previous Post

Next Post >>

(Login to Rate and Mark)

2.8 Average Difficulty : **2.8/5.0**
Based on **49** vote(s)

🟩 Add to TODO List

🟩 Mark as DONE

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

Load Comments