GeeksforGeeks
A computer science portal for geeks

Placements   Practice   GATE CS   IDE   Q&A
GeeksQuiz

Bottom View of a Binary Tree

Given a Binary Tree, we need to print the bottom view from left to right. A node x is there in output if x is the bottommost node at its horizontal distance. Horizontal distance of left child of a node x is equal to horizontal distance of x minus 1, and that of right child is horizontal distance of x plus 1.

Examples:

```
            20
           /    \
          8      22
         / \      \
        5   3      25
           / \
          10  14
```
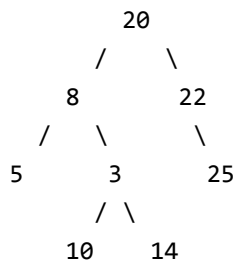
For the above tree the output should be 5, 10, 3, 14, 25.

If there are multiple bottom-most nodes for a horizontal distance from root, then print the later one in level traversal. For example, in the below diagram, 3 and 4 are both the bottom-most nodes at horizontal distance 0, we need to print 4.

```
            20
           /    \
          8      22
         / \    / \
        5   3 4    25
           / \
          10  14
```
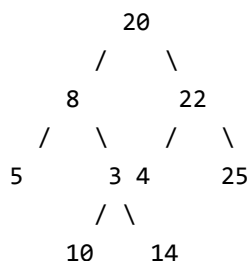
For the above tree the output should be 5, 10, 4, 14, 25.

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

The following are steps to print Bottom View of Bianry Tree.

1. We put tree nodes in a queue for the level order traversal.

2. Start with the horizontal distance(hd) 0 of the root node, keep on adding left child to queue along with the horizontal distance as hd-1 and right child as hd+1.

3. Also, use a TreeMap which stores key value pair sorted on key.

4. Every time, we encounter a new horizontal distance or an existing horizontal distance put the node data for the horizontal distance as key. For the first time it will add to the map, next time it will replace the value. This will make sure that the bottom most element for that horizontal distance is present in the map and if you see the tree from beneath that you will see that element.

A Java based implementation is below :

```java
// Java Program to print Bottom View of Binary Tree
import java.util.*;
import java.util.Map.Entry;

// Tree node class
class TreeNode
{
    int data; //data of the node
    int hd; //horizontal distance of the node
    TreeNode left, right; //left and right references

    // Constructor of tree node
    public TreeNode(int key)
    {
        data = key;
        hd = Integer.MAX_VALUE;
        left = right = null;
    }
}

//Tree class
class Tree
{
    TreeNode root; //root node of tree

    // Default constructor
    public Tree() {}

    // Parameterized tree constructor
    public Tree(TreeNode node)
    {
        root = node;
    }

    // Method that prints the bottom view.
    public void bottomView()
    {
        if (root == null)
            return;

        // Initialize a variable 'hd' with 0 for the root element.
        int hd = 0;

        // TreeMap which stores key value pair sorted on key value
        Map<Integer, Integer> map = new TreeMap<>();

         // Queue to store tree nodes in level order traversal
        Queue<TreeNode> queue = new LinkedList<TreeNode>();
```

```java
        // Assign initialized horizontal distance value to root
        // node and add it to the queue.
        root.hd = hd;
        queue.add(root);

        // Loop until the queue is empty (standard level order loop)
        while (!queue.isEmpty())
        {
            TreeNode temp = queue.remove();

            // Extract the horizontal distance value from the
            // dequeued tree node.
            hd = temp.hd;

            // Put the dequeued tree node to TreeMap having key
            // as horizontal distance. Every time we find a node
            // having same horizontal distance we need to replace
            // the data in the map.
            map.put(hd, temp.data);

            // If the dequeued node has a left child add it to the
            // queue with a horizontal distance hd-1.
            if (temp.left != null)
            {
                temp.left.hd = hd-1;
                queue.add(temp.left);
            }
            // If the dequeued node has a left child add it to the
            // queue with a horizontal distance hd+1.
            if (temp.right != null)
            {
                temp.right.hd = hd+1;
                queue.add(temp.right);
            }
        }

        // Extract the entries of map into a set to traverse
        // an iterator over that.
        Set<Entry<Integer, Integer>> set = map.entrySet();

        // Make an iterator
        Iterator<Entry<Integer, Integer>> iterator = set.iterator();

        // Traverse the map elements using the iterator.
        while (iterator.hasNext())
        {
            Map.Entry<Integer, Integer> me = iterator.next();
            System.out.print(me.getValue()+" ");
        }
    }
}

// Main driver class
public class BottomView
{
    public static void main(String[] args)
    {
        TreeNode root = new TreeNode(20);
        root.left = new TreeNode(8);
        root.right = new TreeNode(22);
        root.left.left = new TreeNode(5);
        root.left.right = new TreeNode(3);
        root.right.left = new TreeNode(4);
        root.right.right = new TreeNode(25);
```

```
        root.left.right.left = new TreeNode(10);
        root.left.right.right = new TreeNode(14);
        Tree tree = new Tree(root);
        System.out.println("Bottom view of the given binary tree:");
        tree.bottomView();
    }
}
```

Run on IDE

Output:

```
Bottom view of the given binary tree:
5 10 4 14 25
```

Exercise: Extend the above solution to print all bottommost nodes at a horizontal distance if there are multiple bottommost nodes. For the above second example, the output should be 5 10 3 4 14 25.

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

81 Comments Category: Trees

Related Posts:

- 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?

- Closest leaf to a given node in Binary Tree

(Login to Rate and Mark)

3.8   Average Difficulty : **3.8/5.0**
Based on **6** vote(s)

☐ Add to TODO List

☐ Mark as DONE

Like   Share   16 people like this.

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

@geeksforgeeks, Some rights reserved   Contact Us!   About Us!   Advertise with us!