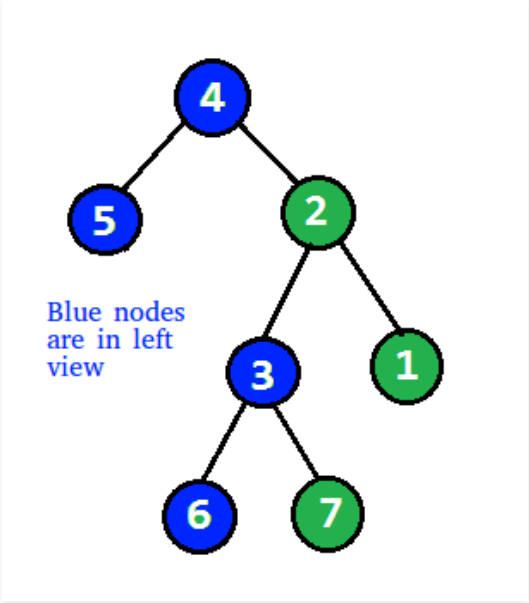
Print Left View of a Binary Tree

Difficulty Level : Medium • Last Updated : 23 May, 2021

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



Examples:

```
Input :
                  1
               /    \
              2      3
             / \      \
            4   5      6
Output : 1 2 4

Input :
         1
       /   \
     2       3
       \
         4
```



```
Output :1 2 4 5 6
```

Recommended: Please solve it on "***PRACTICE***" first, before moving on to the solution.

Method-1 (Using Recursion)

The left view contains all nodes that are first nodes in their levels. A simple solution is to **do level order traversal**

and print the first node in every level.

The problem can also be solved **using simple recursive traversal**. We can keep track of the level of a node by passing a parameter to all recursive calls. The idea is to keep track of the maximum level also. 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 left subtree before right subtree).

Below is the implementation of the above idea-

C++

```cpp
// C++ program to print left view of Binary Tree
#include <bits/stdc++.h>
using namespace std;

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
// left view of a binary tree.
void leftViewUtil(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)
    {
        cout << root->data << " ";
        *max_level = level;
    }

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

}

// A wrapper over leftViewUtil()
void leftView(struct Node *root)
```

```
{
    int max_level = 0;
    leftViewUtil(root, 1, &max_level);
}

// Driver Code
int main()
{
    Node* root = newNode(10);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(7);
    root->left->right = newNode(8);
    root->right->right = newNode(15);
    root->right->left = newNode(12);
    root->right->right->left = newNode(14);

    leftView(root);

    return 0;
}
```

C

```
// C program to print left 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 left view of a binary tree.
void leftViewUtil(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 left and right subtrees
    leftViewUtil(root->left, level + 1, max_level);
    leftViewUtil(root->right, level + 1, max_level);
}

// A wrapper over leftViewUtil()
void leftView(struct node* root)
{
    int max_level = 0;
    leftViewUtil(root, 1, &max_level);
}

// Driver code
int main()
{
```

```
    struct node* root = newNode(12);
    root->left = newNode(10);
    root->right = newNode(30);
    root->right->left = newNode(25);
    root->right->right = newNode(40);

    leftView(root);

    return 0;
}
```

Java

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

/* Class containing left and right child of current
node and key value*/
class Node {
    int data;
    Node left, right;

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

/* Class to print the left view */
class BinaryTree {
    Node root;
    static int max_level = 0;

    // recursive function to print left view
    void leftViewUtil(Node node, int level)
```

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

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

            // Recur for left and right subtrees
            leftViewUtil(node.left, level + 1);
            leftViewUtil(node.right, level + 1);
        }

        // A wrapper over leftViewUtil()
        void leftView()
        {
            leftViewUtil(root, 1);
        }

        /* testing for example nodes */
        public static void main(String args[])
        {
            /* creating a binary tree and entering the nodes */
            BinaryTree tree = new BinaryTree();
            tree.root = new Node(12);
            tree.root.left = new Node(10);
            tree.root.right = new Node(30);
            tree.root.right.left = new Node(25);
            tree.root.right.right = new Node(40);

            tree.leftView();
        }
    }
```

Python

```python
# Python program to print left view of Binary Tree

# A binary tree node
class Node:

    # Constructor to create a new node
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None


# Recursive function pritn left view of a binary tree
def leftViewUtil(root, level, max_level):

    # Base Case
    if root is None:
        return

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

    # Recur for left and right subtree
    leftViewUtil(root.left, level + 1, max_level)
    leftViewUtil(root.right, level + 1, max_level)


# A wrapper over leftViewUtil()
def leftView(root):
    max_level = [0]
    leftViewUtil(root, 1, max_level)
```

```
# Driver program to test above function
root = Node(12)
root.left = Node(10)
root.right = Node(20)
root.right.left = Node(25)
root.right.right = Node(40)

leftView(root)

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

C#

```
using System;

// C# program to print left view of binary tree

/* Class containing left and right child of current
node and key value*/
public class Node {
    public int data;
    public Node left, right;

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

/* Class to print the left view */
public class BinaryTree {
    public Node root;
    public static int max_level = 0;
```

```
// recursive function to print left view
public virtual void leftViewUtil(Node node, int level)
{
    // Base Case
    if (node == null) {
        return;
    }

    // If this is the first node of its level
    if (max_level < level) {
        Console.Write(" " + node.data);
        max_level = level;
    }

    // Recur for left and right subtrees
    leftViewUtil(node.left, level + 1);
    leftViewUtil(node.right, level + 1);
}

// A wrapper over leftViewUtil()
public virtual void leftView()
{
    leftViewUtil(root, 1);
}

/* testing for example nodes */
public static void Main(string[] args)
{
    /* creating a binary tree and entering the nodes */
    BinaryTree tree = new BinaryTree();
    tree.root = new Node(12);
    tree.root.left = new Node(10);
    tree.root.right = new Node(30);
    tree.root.right.left = new Node(25);
    tree.root.right.right = new Node(40);
```

```
            tree.leftView();
        }
    }

    // This code is contributed by Shrikant13
```

Output

```
1    2    4    8
```

Time Complexity: The function does a simple traversal of the tree, so the complexity is O(n).
Auxiliary Space: O(n), due to the stack space during recursive call.

Method-2 (Using Queue):

In this method, level order traversal based solution is discussed. If we observe carefully, we will see that our main task is to print the left most node of every level. So, we will do a level order traversal on the tree and print the leftmost node at every level. Below is the implementation of above approach:

C++

```
// C++ program to print left view of
// Binary Tree

#include<bits/stdc++.h>
using namespace std;

// A Binary Tree Node
struct Node
{
    int data;
```

```
    struct Node *left, *right;
};

// Utility function to create a new tree node
Node* newNode(int data)
{
    Node *temp = new Node;
    temp->data = data;
    temp->left = temp->right = NULL;
    return temp;
}

// function to print left view of
// binary tree
void printLeftView(Node* root)
{
    if (!root)
        return;

    queue<Node*> q;
    q.push(root);

    while (!q.empty())
    {
        // number of nodes at current level
        int n = q.size();

        // Traverse all nodes of current level
        for(int i = 1; i <= n; i++)
        {
            Node* temp = q.front();
            q.pop();

            // Print the left most element
            // at the level
            if (i == 1)
                cout<<temp->data<<" ";
```

```
            // Add left node to queue
            if (temp->left != NULL)
                q.push(temp->left);

            // Add right node to queue
            if (temp->right != NULL)
                q.push(temp->right);
        }
    }
}

// Driver code
int main()
{
    // Let's construct the tree as
    // shown in example

    Node* root = newNode(10);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(7);
    root->left->right = newNode(8);
    root->right->right = newNode(15);
    root->right->left = newNode(12);
    root->right->right->left = newNode(14);

    printLeftView(root);
}

// This code is contributed by
// Manne SreeCharan
```

Java

```java
// Java program to print left view of Binary
// Tree
import java.util.*;

public class PrintRightView {
    // Binary tree node
    private static class Node {
        int data;
        Node left, right;

        public Node(int data)
        {
            this.data = data;
            this.left = null;
            this.right = null;
        }
    }

    // function to print left view of binary tree
    private static void printLeftView(Node root)
    {
        if (root == null)
            return;

        Queue<Node> queue = new LinkedList<>();
        queue.add(root);

        while (!queue.isEmpty()) {
            // number of nodes at current level
            int n = queue.size();

            // Traverse all nodes of current level
            for (int i = 1; i <= n; i++) {
                Node temp = queue.poll();

                // Print the left most element at
                // the level
```

```
                    if (i == 1)
                        System.out.print(temp.data + " ");

                    // Add left node to queue
                    if (temp.left != null)
                        queue.add(temp.left);

                    // Add right node to queue
                    if (temp.right != null)
                        queue.add(temp.right);
                }
            }
        }

        // Driver code
        public static void main(String[] args)
        {
            // construct binary tree as shown in
            // above diagram
            Node root = new Node(10);
            root.left = new Node(2);
            root.right = new Node(3);
            root.left.left = new Node(7);
            root.left.right = new Node(8);
            root.right.right = new Node(15);
            root.right.left = new Node(12);
            root.right.right.left = new Node(14);

            printLeftView(root);
        }
    }

    // This code is contributed by
    // Manne SreeCharan
```

Python

```python
# Python3 program to print left view of
# Binary Tree

# Binary Tree Node
""" utility that allocates a newNode
with the given key """


class newNode:

    # Construct to create a newNode
    def __init__(self, key):
        self.data = key
        self.left = None
        self.right = None
        self.hd = 0

# function to print left view of
# binary tree


def printRightView(root):

    if (not root):
        return

    q = []
    q.append(root)

    while (len(q)):

        # number of nodes at current level
        n = len(q)
```

```
        # Traverse all nodes of current level
        for i in range(1, n + 1):
            temp = q[0]
            q.pop(0)

            # Print the left most element
            # at the level
            if (i == 1):
                print(temp.data, end=" ")

            # Add left node to queue
            if (temp.left != None):
                q.append(temp.left)

            # Add right node to queue
            if (temp.right != None):
                q.append(temp.right)


# Driver Code
if __name__ == '__main__':

    root = newNode(10)
    root.left = newNode(2)
    root.right = newNode(3)
    root.left.left = newNode(7)
    root.left.right = newNode(8)
    root.right.right = newNode(15)
    root.right.left = newNode(12)
    root.right.right.left = newNode(14)
    printRightView(root)

# This code is contributed by
# Manne SreeCharan
```

C#

```
// C# program to print left view
// of Binary Tree
using System;
using System.Collections.Generic;

public class PrintRightView {
    // Binary tree node
    private class Node {
        public int data;
        public Node left, right;

        public Node(int data)
        {
            this.data = data;
            this.left = null;
            this.right = null;
        }
    }

    // function to print left view of binary tree
    private static void printRightView(Node root)
    {
        if (root == null)
            return;

        Queue<Node> queue = new Queue<Node>();
        queue.Enqueue(root);

        while (queue.Count != 0) {
            // number of nodes at current level
            int n = queue.Count;

            // Traverse all nodes of current level
            for (int i = 1; i <= n; i++) {
```

```
                    Node temp = queue.Dequeue();

                    // Print the left most element at
                    // the level
                    if (i == n)
                        Console.Write(temp.data + " ");

                    // Add left node to queue
                    if (temp.left != null)
                        queue.Enqueue(temp.left);

                    // Add right node to queue
                    if (temp.right != null)
                        queue.Enqueue(temp.right);
                }
            }
        }

        // Driver code
        public static void Main(String[] args)
        {
            // construct binary tree as shown in
            // above diagram
            Node root = new Node(10);
            root.left = new Node(2);
            root.right = new Node(3);
            root.left.left = new Node(7);
            root.left.right = new Node(8);
            root.right.right = new Node(15);
            root.right.left = new Node(12);
            root.right.right.left = new Node(14);
            printRightView(root);
        }
    }

    // This code is contributed Manne SreeCharan
```

Output

```
10 2 7 14
```

Time Complexity: O(n), where n is the number of nodes in the binary tree.

This article is contributed by Ramsai Chinthamani, Manne SreeCharan Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Attention reader! Don't stop learning now. Get hold of all the important DSA concepts with the **DSA Self Paced Course** at a student-friendly price and become industry ready. To complete your preparation from learning a language to DS Algo and many more, please refer **Complete Interview Preparation Course**.

In case you wish to attend live classes with industry experts, please refer **Geeks Classes Live**

Next >

Print Right View of a Binary Tree

RECOMMENDED ARTICLES

Page : 1 2 3

01 Iterative Method To Print Left View of a Binary Tree
01, Feb 19

02 Sum of nodes in the left view of the given binary tree
20, Sep 19

03 Check if the Left View of the given tree is sorted or not
19, May 20

04 Print Right View of a Binary Tree
05, Apr 14

05 Print Nodes in Top View of Binary Tree
07, Nov 14

06 Print nodes in top view of Binary Tree | Set 2
11, Sep 18

07 Print nodes in the Top View of Binary Tree | Set 3
30, Oct 18

08 Print Bottom-Right View of a Binary Tree
14, Aug 19

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy | Normal | Medium | Hard | Expert

Improved By : shrikanth13, rathbhupendra, hunter2000, sukhoi33, gnitish31, AakashYadav4, Rakesh Kumar Mallick

Article Tags : Accolite, Amazon, Flipkart, Knowlarity, MakeMyTrip, Ola Cabs, Open Solutions, Paytm, Qualcomm, Samsung, Snapdeal, tree-view, Twitter, Tree

Practice Tags : Paytm, Flipkart, Accolite, Amazon, Samsung, Snapdeal, MakeMyTrip, Ola Cabs, Qualcomm, Twitter, Knowlarity, Open Solutions, Tree

Improve Article | Report Issue

Load Comments

GeeksforGeeks
5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org