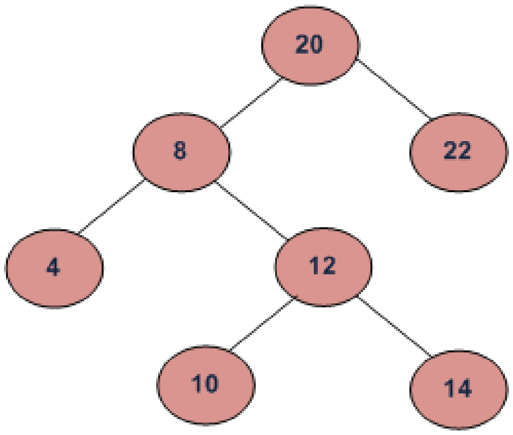
Print nodes between two given level numbers of a binary tree

Given a binary tree and two level numbers 'low' and 'high', print nodes from level low to level high.

```
For example consider the binary tree given in below diagram.

Input: Root of below tree, low = 2, high = 4

Output:
8 22
4 12
10 14
```



A **Simple Method** is to first write a recursive function that prints nodes of a given level number. Then call recursive function in a loop from low to high. Time complexity of this method is O(n^2)
We can print nodes **in O(n) time** using queue based iterative level order traversal. The idea is to do simple queue based level order traversal. While doing inorder traversal, add a marker node at the end. Whenever we see a marker node, we increase level number. If level number is between low and high, then print nodes.

The following is the implementation of above idea.

C++

```
// A C++ program to print Nodes level by level berween given two levels.
#include <iostream>
#include <queue>
using namespace std;

/* A binary tree Node has key, pointer to left and right children */
struct Node
{
    int key;
    struct Node* left, *right;
};

/* Given a binary tree, print nodes from level number 'low' to level
   number 'high'*/
void printLevels(Node* root, int low, int high)
{
    queue <Node *> Q;

    Node *marker = new Node; // Marker node to indicate end of level

    int level = 1;   // Initialize level number

    // Enqueue the only first level node and marker node for end of level
    Q.push(root);
    Q.push(marker);

    // Simple level order traversal loop
    while (Q.empty() == false)
    {
        // Remove the front item from queue
        Node *n = Q.front();
        Q.pop();

        // Check if end of level is reached
        if (n == marker)
        {
            // print a new line and increment level number
            cout << endl;
            level++;

            // Check if marker node was last node in queue or
            // level number is beyond the given upper limit
            if (Q.empty() == true || level > high) break;

            // Enqueue the marker for end of next level
            Q.push(marker);

            // If this is marker, then we don't need print it
            // and enqueue its children
            continue;
        }

        // If level is equal to or greater than given lower level,
        // print it
        if (level >= low)
            cout << n->key << " ";

        // Enqueue children of non-marker node
        if (n->left != NULL)  Q.push(n->left);
        if (n->right != NULL) Q.push(n->right);
    }
}

/* Helper function that allocates a new Node with the
   given key and NULL left and right pointers. */
Node* newNode(int key)
{
    Node* temp = new Node;
    temp->key = key;
    temp->left = temp->right = NULL;
    return (temp);
}

/* Driver program to test above functions*/
int main()
{
    // Let us construct the BST shown in the above figure
    struct Node *root        = newNode(20);
    root->left               = newNode(8);
    root->right              = newNode(22);
    root->left->left         = newNode(4);
    root->left->right        = newNode(12);
    root->left->right->left  = newNode(10);
    root->left->right->right = newNode(14);

    cout << "Level Order traversal between given two levels is";
    printLevels(root, 2, 3);

    return 0;
}
```

Run on IDE

Java

```
// Java program to print Nodes level by level between given two levels
import java.util.LinkedList;
import java.util.Queue;

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

    int data;
    Node left, right;
    // Used to indicate whether the right pointer is a normal
    // right pointer or a pointer to inorder successor.
    boolean isThreaded;

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

class BinaryTree {

    static Node root;

    /* Given a binary tree, print nodes from level number 'low' to level
     number 'high'*/
    void printLevels(Node node, int low, int high) {
        Queue<Node> Q = new LinkedList<>();
        Node  marker = new Node(4); // Marker node to indicate end of level

        int level = 1;   // Initialize level number

        // Enqueue the only first level node and marker node for end of level
        Q.add(node);
        Q.add(marker);

        // Simple level order traversal loop
        while (Q.isEmpty() == false) {
            // Remove the front item from queue
            Node  n = Q.peek();
            Q.remove();

            // Check if end of level is reached
            if (n == marker) {
                // print a new line and increment level number
                System.out.println("");
                level++;

                // Check if marker node was last node in queue or
                // level number is beyond the given upper limit
                if (Q.isEmpty() == true || level > high) {
                    break;
                }

                // Enqueue the marker for end of next level
                Q.add(marker);

                // If this is marker, then we don't need print it
                // and enqueue its children
                continue;
            }

            // If level is equal to or greater than given lower level,
            // print it
            if (level >= low) {
                System.out.print( n.data + " ");
            }

            // Enqueue children of non-marker node
            if (n.left != null) {
                Q.add(n.left);
            }
            if (n.right != null) {
                Q.add(n.right);
            }
        }
    }

    // driver program to test for above functions
    public static void main(String args[]) {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(20);
        tree.root.left = new Node(8);
        tree.root.right = new Node(22);

        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(12);
        tree.root.left.right.left = new Node(10);
        tree.root.left.right.right = new Node(14);

        System.out.print("Level Order traversal between given two levels is ");
        tree.printLevels(root, 2, 3);
    }
}

// This code has been contributed by Mayank Jaiswal
```

Run on IDE

Python

```
# Python program to print nodes level by level betweeen
# given two levels

# A binary tree node
class Node:
    # Constructor tor create a new node
    def __init__(self, key):
        self.key = key
        self.left = None
        self.right = None

# Given a binary tree, print nodes form level number 'low'
# to level number 'high'

def printLevels(root, low, high):
    Q = []

    marker  = Node(11114) # Marker node to indicate end of level

    level = 1 # Initialize level number

    # Enqueue the only first level node and marker node for
    # end of level
    Q.append(root)
    Q.append(marker)

    #print Q
    # Simple level order traversal loop
    while(len(Q) >0):
        # Remove the front item from queue
        n = Q[0]
        Q.pop(0)
        #print Q
        # Check if end of level is reached
        if n == marker:
            # print a new line and increment level number
            print
            level += 1

            # Check if marker node was last node in queue
            # or level nubmer is beyond the given upper limit
            if len(Q) == 0 or level > high:
                break

            # Enqueue the marker for end of next level
            Q.append(marker)

            # If this is marker, then we don't need print it
            # and enqueue its children
            continue
        if level >= low:
                print n.key,
        # Enqueue children of non-marker node
        if n.left is not None:
            Q.append(n.left)
            Q.append(n.right)

# Driver program to test the above function
root = Node(20)
root.left = Node(8)
root.right = Node(22)
root.left.left = Node(4)
root.left.right = Node(12)
root.left.right.left = Node(10)
root.left.right.right = Node(14)

print "Level Order Traversal between given two levels is",
printLevels(root,2,3)

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

Run on IDE

```
Level Order traversal between given two levels is
8 22
4 12
```

Time complexity of above method is O(n) as it does a simple level order traversal.

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

Learn to Program in Python
Python for Data Science
A free online course from Microsoft
edX www.edx.org
Enroll Now

36 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

Like Share 12 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!