Print level order traversal line by line

Given a binary tree, print level order traversal in a way that nodes of all levels are printed in separate lines.

For example consider the following tree

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

Output for above tree should be
1
2 3
4 5 6
7 8 9
```

Note that this is different from simple level order traversal where we need to print all nodes together. Here we need to print nodes of different levels in different lines.

A simple solution is to print use the recursive function discussed in the level order traversal post and print a new line after every call to printGivenLevel().

```
/* Function to line by line print level order traversal a tree*/
void printLevelOrder(struct node* root)
{
    int h = height(root);
    int i;
    for (i=1; i<=h; i++)
    {
        printGivenLevel(root, i);
        printf("\n");
    }
}

/* Print nodes at a given level */
void printGivenLevel(struct node* root, int level)
{
    if (root == NULL)
        return;
    if (level == 1)
        printf("%d ", root->data);
    else if (level > 1)
    {
        printGivenLevel(root->left, level-1);
        printGivenLevel(root->right, level-1);
    }
}
```

The time complexity of the above solution is O(n^2)

How to modify the iterative level order traversal (Method 2 of this) to levels line by line?

The idea is similar to this post. We count the nodes at current level. And for every node, we enqueue its children to queue.

```
/* Iterative program to print levels line by line */
#include <iostream>
#include <queue>
using namespace std;

// A Binary Tree Node
struct node
{
    struct node *left;
    int data;
    struct node *right;
    struct node *right;
};

// Iterative method to do level order traversal line by line
void printLevelOrder(node *root)
{
    // Base Case
    if (root == NULL)  return;

    // Create an empty queue for level order tarversal
    queue<node *> q;

    // Enqueue Root and initialize height
    q.push(root);

    while (1)
    {
        // nodeCount (queue size) indicates number of nodes
        // at current lelvel.
        int nodeCount = q.size();
        if (nodeCount == 0)
            break;

        // Dequeue all nodes of current level and Enqueue all
        // nodes of next level
        while (nodeCount > 0)
        {
            node *node = q.front();
            cout << node->data << " ";
            q.pop();
            if (node->left != NULL)
                q.push(node->left);
            if (node->right != NULL)
                q.push(node->right);
            nodeCount--;
        }
        cout << endl;
    }
}

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

// Driver program to test above functions
int main()
{
    // Let us create binary tree shown in above diagram
    node *root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);
    root->right->right = newNode(6);

    printLevelOrder(root);
    return 0;
}
```

Output:

```
1
2 3
4 5 6
```

Time complexity of this method is O(n) where n is number of nodes in given binary tree.