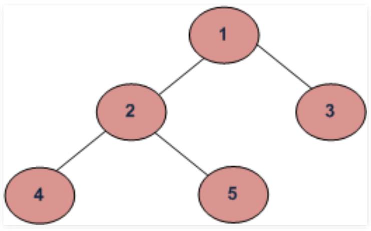
Level Order Tree Traversal

Level order traversal of a tree is breadth first traversal for the tree.



Example Tree

Level order traversal of the above tree is 1 2 3 4 5

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

METHOD 1 (Use function to print a given level)

Algorithm:

There are basically two functions in this method. One is to print all nodes at a given level (printGivenLevel), and other is to print level order traversal of the tree (printLevelorder). printLevelorder makes use of printGivenLevel to print nodes at all levels one by one starting from root.

```
/*Function to print level order traversal of tree*/
printLevelorder(tree)
for d = 1 to height(tree)
   printGivenLevel(tree, d);

/*Function to print all nodes at a given level*/
printGivenLevel(tree, level)
if tree is NULL then return;
if level is 1, then
    print(tree->data);
else if level greater than 1, then
    printGivenLevel(tree->left, level-1);
    printGivenLevel(tree->right, level-1);
```

Implementation:

C   Java   Python

```
# Recursive Python program for level order

# A node structure
class Node:

    # A utility function to create a new r
    def __init__(self, key):
        self.data = key
        self.left = None
        self.right = None


# Function to  print level order traversal
def printLevelOrder(root):
    h = height(root)
    for i in range(1, h+1):
        printGivenLevel(root, i)


# Print nodes at a given level
def printGivenLevel(root , level):
    if root is None:
        return
    if level == 1:
        print "%d" %(root.data),
    elif level > 1 :
        printGivenLevel(root.left , level-
        printGivenLevel(root.right , level


""" Compute the height of a tree--the numb
    along the longest path from the root r
    the farthest leaf node
"""
def height(node):
    if node is None:
        return 0
    else :
        # Compute the height of each subtr
        lheight = height(node.left)
        rheight = height(node.right)
        #Use the larger one
        if lheight > rheight :
            return lheight+1
        else:
            return rheight+1

# Driver program to test above function
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)

print "Level order traversal of binary tre
printLevelOrder(root)

#This code is contributed by Nikhil Kumar
```

Run on IDE

Output:

```
Level order traversal of binary tree is -
1 2 3 4 5
```

Time Complexity: O(n^2) in worst case. For a skewed tree, printGivenLevel() takes O(n) time where n is the number of nodes in the skewed tree. So time complexity of printLevelOrder() is O(n) + O(n-1) + O(n-2) + .. + O(1) which is O(n^2).

METHOD 2 (Use Queue)

Algorithm:

For each node, first the node is visited and then it's child nodes are put in a FIFO queue.

```
printLevelorder(tree)
1) Create an empty queue q
2) temp_node = root /*start from root*/
3) Loop while temp_node is not NULL
    a) print temp_node->data.

    b) Enqueue temp_node's children (first left then right children) to q
    c) Dequeue a node from q and assign it's value to temp_node
```

Implementation:

Here is a simple implementation of the above algorithm. Queue is implemented using an array with maximum size of 500. We can implement queue as linked list also.

C   C++   Java   Python

```
# Python program to print level order trav

# A node structure
class Node:
    # A utility function to create a new r
    def __init__(self ,key):
        self.data = key
        self.left = None
        self.right = None

# Iterative Method to print the height of
def printLevelOrder(root):
    # Base Case
    if root is None:
        return

    # Create an empty queue for level orde
    queue = []

    # Enqueue Root and initialize height
    queue.append(root)

    while(len(queue) > 0):
        # Print front of queue and remove
        print queue[0].data,
        node = queue.pop(0)

        #Enqueue left child
        if node.left is not None:
            queue.append(node.left)
        # Enqueue right child
        if node.right is not None:
            queue.append(node.right)

#Driver Program to test above function
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)

print "Level Order Traversal of binary tre
printLevelOrder(root)
#This code is contributed by Nikhil Kumar
```

Run on IDE

Output:

```
Level order traversal of binary tree is -
1 2 3 4 5
```

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

Asked in: Amazon, Cisco, DE Shaw, Flipkart, Microsoft, Payu

References:

http://en.wikipedia.org/wiki/Breadth-first_traversal

Please write comments if you find any bug in the above programs/algorithms or other ways to solve the same problem.