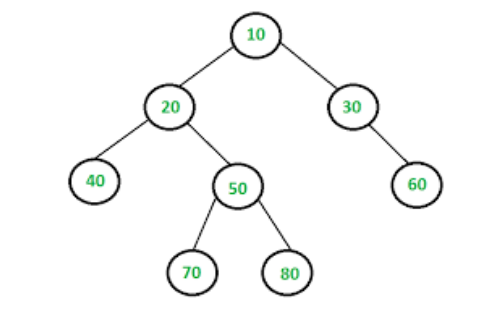
***Level Order Traversal of a Binary Tree***

We have seen the three basic traversals(Preorder, postorder, and Inorder) of a Binary Tree. We can also traverse a Binary Tree using the *Level Order Traversal*.  
  
In the Level Order Traversal, the binary tree is traversed level-wise starting from the first to last level sequentially.  
  
Consider the below binary tree:  
  
  
The Level Order Traversal of the above Binary Tree will be: **10 20 30 40 50 60 70 80**.  
  
**Algorithm**: The Level Order Traversal can be implemented efficiently using a Queue.

1. Create an empty queue q.
2. Push the root node of tree to q. That is, q.push(root).
3. Loop while the queue is not empty:
   * Pop the top node from queue and print the node.
   * Enqueue node's children (first left then right children) to q
   * Repeat the process until queue is not empty.

**Implementation**:  
C++Java

// Iterative Queue based Java program to do

// level order traversal of Binary Tree

import java.util.Queue;

import java.util.LinkedList;

/\* Class to represent Tree node \*/

class Node {

int data;

Node left, right;

public Node(int item) {

data = item;

left = null;

right = null;

}

}

/\* Class to print Level Order Traversal \*/

class BinaryTree {

Node root;

/\* Given a binary tree. Print its nodes in

level order using array for implementing queue \*/

void printLevelOrder()

{

Queue<Node> queue = new LinkedList<Node>();

queue.add(root);

while (!queue.isEmpty())

{

Node tempNode = queue.poll();

System.out.print(tempNode.data + " ");

/\* Enqueue left child \*/

if (tempNode.left != null) {

queue.add(tempNode.left);

}

/\* Enqueue right child \*/

if (tempNode.right != null) {

queue.add(tempNode.right);

}

}

}

// Driver Code

public static void main(String args[])

{

// Create the following Binary Tree

// 1

// / \

// 2 3

// / \

// 4 5

BinaryTree tree\_level = new BinaryTree();

tree\_level.root = new Node(1);

tree\_level.root.left = new Node(2);

tree\_level.root.right = new Node(3);

tree\_level.root.left.left = new Node(4);

tree\_level.root.left.right = new Node(5);

System.out.println("Level order traversal " +

"of binary tree is - ");

tree\_level.printLevelOrder();

}

}

**Output**:

1 2 3 4 5

**Time Complexity**: O(N), where N is the number of nodes in the Tree.  
**Auxiliary Space**: O(N)