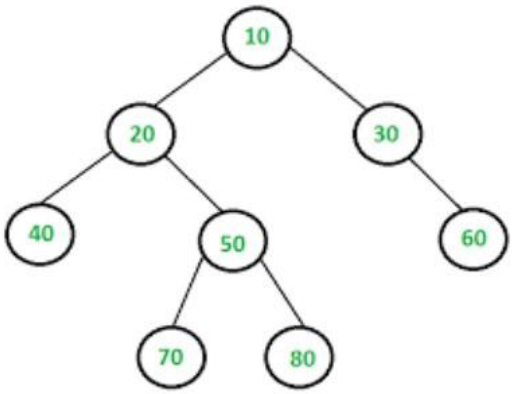
Level order Traversal of Binary Trees

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

```cpp
// C++ program to print level order traversal of a Tree
#include <iostream>
#include <queue>
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 Level Order Traversal of the Binary Tree
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 (q.empty() == false)
    {
        // Print front of queue and remove it from queue
        Node *node = q.front();
        cout << node->data << " ";
        q.pop();
        /* Enqueue left child */
        if (node->left != NULL)
            q.push(node->left);
        /* Enqueue right child */
        if (node->right != NULL)
            q.push(node->right);
    }
}
// Driver Code
int main()
{
    // Create the following Binary Tree
    //      1
    //     / \
    //    2   3
    //   / \
    //  4   5

    Node *root = newNode(1);
```

```
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);

    cout << "Level Order traversal of binary tree is \n";
    printLevelOrder(root);

    return 0;
}
```

Output:

```
1 2 3 4 5
```

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