```
int findMaxLen(Node root) {
    int nMaxLen = 0;
    if (root == null)
        return 0;
    if (root.left == null)
        root.nMaxLeft = 0;
    if (root.right == null)
        root.nMaxRight = 0;
    if (root.left != null)
        findMaxLen(root.left);
    if (root.right != null)
        findMaxLen(root.right);
    if (root.left != null) {
        int nTempMaxLen = 0;
        nTempMaxLen = (root.left.nMaxLeft > root.left.nMaxRight) ?
            root.left.nMaxLeft : root.left.nMaxRight;
        root.nMaxLeft = nTempMaxLen + 1;
    }
    if (root.right != null) {
        int nTempMaxLen = 0;
        nTempMaxLen = (root.right.nMaxLeft > root.right.nMaxRight) ?
            root.right.nMaxLeft : root.right.nMaxRight;
        root.nMaxRight = nTempMaxLen + 1;
    }
    if (root.nMaxLeft + root.nMaxRight > nMaxLen)
        nMaxLen = root.nMaxLeft + root.nMaxRight;
    return nMaxLen;
}
```

Time Complexity: O(*n*). Space Complexity: O(*n*).

**Problem-19** Give an algorithm for finding the level that has the maximum sum in the binary tree.

**Solution:** The logic is very much similar to finding the number of levels. The only change is, we