Maximum Depth of Binary Tree

Given a binary tree, find its maximum depth.

The maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

Solution 1

After a solution is accepted it would be very helpful to know how to make it run faster looking at better performing solution(s).

written by rainhacker original link here

Solution 2

if the node does not exist, simply return 0. Otherwise, return the 1+the longer distance of its subtree.

```
public int maxDepth(TreeNode root) {
        if(root==null){
            return 0;
        }
        return 1+Math.max(maxDepth(root.left),maxDepth(root.right));
    }
```

written by ray050899 original link here

Solution 3

1. Depth-first-search

Only one line code.

```
int maxDepth(TreeNode *root)
{
    return root == NULL ? 0 : max(maxDepth(root -> left), maxDepth(root -> right)
) + 1;
}
```

2. Breadth-first-search

Calculate the count of the last level.

```
int maxDepth(TreeNode *root)
{
    if(root == NULL)
        return 0;

    int res = 0;
    queue<TreeNode *> q;
    q.push(root);
    while(!q.empty())
    {
        ++ res;
        for(int i = 0, n = q.size(); i < n; ++ i)
        {
            TreeNode *p = q.front();
            q.pop();

            if(p -> left != NULL)
                q.push(p -> left);
            if(p -> right != NULL)
                q.push(p -> right);
        }
    }

    return res;
}
```

written by makuiyu original link here