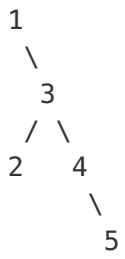
Binary Tree Longest Consecutive Sequence

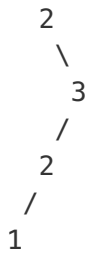
Given a binary tree, find the length of the longest consecutive sequence path.

The path refers to any sequence of nodes from some starting node to any node in the tree along the parent-child connections. The longest consecutive path need to be from parent to child (cannot be the reverse).

For example,



Longest consecutive sequence path is `3-4-5`, so return `3`.



Longest consecutive sequence path is `2-3`,not `3-2-1`, so return `2`.

Solution 1

Just very intuitive depth-first search, send cur node value to the next level and compare it with the next level node.

```
public class Solution {
    private int max = 0;
    public int longestConsecutive(TreeNode root) {
        if(root == null) return 0;
        helper(root, 0, root.val);
        return max;
    }

    public void helper(TreeNode root, int cur, int target){
        if(root == null) return;
        if(root.val == target) cur++;
        else cur = 1;
        max = Math.max(cur, max);
        helper(root.left, cur, root.val + 1);
        helper(root.right, cur, root.val + 1);
    }
}
```

written by czonzhu original link here

Solution 2

```
class Solution {
public:
    int longestConsecutive(TreeNode* root) {
        return search(root, nullptr, 0);
    }

    int search(TreeNode *root, TreeNode *parent, int len) {
        if (!root) return len;
        len = (parent && root->val == parent->val + 1)?len+1:1;
        return max(len, max(search(root->left, root, len), search(root->right, ro
ot, len)));
    }
};
```

len stores the longest path till current node.

written by lightmark original link here

Solution 3

```
public class Solution {
    public int longestConsecutive(TreeNode root) {
        return (root==null)?0:Math.max(dfs(root.left, 1, root.val), dfs(root.righ
t, 1, root.val));
    }

    public int dfs(TreeNode root, int count, int val){
        if(root==null) return count;
        count = (root.val - val == 1)?count+1:1;
        int left = dfs(root.left, count, root.val);
        int right = dfs(root.right, count, root.val);
        return Math.max(Math.max(left, right), count);
    }
}
```

written by nightowl original link here

From Leetcoder.