Longest Path of a Binary Tree

Given a binary tree root node, return the number of nodes in the longest path between the root and a leaf node  
  
Input: Node in a Binary Tree  
Output: Integer

# Example

Input: Output: 3 (from path 1 -> 3 -> 4)

# 

# Constraints

Time Complexity: O(N)  
Auxiliary Space Complexity: O(N)

The binary tree node has the following properties:

value : an integer  
leftChild : default null  
rightChild : default null

# Solution

1. Create a scope variable called ‘max’ to keep an integer of the longest path
2. Create a recursive helper method called ‘search’ that takes in a node and length of path
   1. The base case is if the node is null
   2. Otherwise update the ‘max’ if the length of path is larger than max
   3. Perform a recursive case for both the left and right child with the length incremented by one.
3. Invoke the ‘search’ method with the root node and a length of 1
4. When the ‘search’ is complete, return the ‘max’ count

**JavaScript Solution:**

function longestPath(root) {

var max = 0;

function search(node, length) {

if(!node){ return; }

max = Math.max(max, length);

search(node.left, length + 1);

search(node.right, length + 1);

}

search(root, 1);

return max;

}

# Resources

https://leetcode.com/problems/maximum-depth-of-binary-tree/