Problem Name: Largest Distance between nodes of a Tree

**Topics:**

**Companies:**

**Level:** Medium

**Language:** C++

**Problem Statement:**

**Input Format:**

**Output Format:**

**Constraints:**

**Examples:**

**Approach one Solution:**

**Explanation:**

. 1) pick any node u  
2) find the node which is farthest from u, call it x (calculate using the same approach as in Solution 1)  
3) find the node which is farthest from x, call it q (calculate using the same approach as in Solution 1)  
The answer will be the length of a path from x to q.

Proof of correctness:

The crucial step is to prove that x will be one of the endpoints of the path with maximal length (note that there might be more than one such path). If it is, then the longest path from x will be the longest path in the tree.

Let d(v1, v2) be length of path between v1 and v2

Let’s prove it by contradiction: assume there is a strictly longer path between s and t, neither of which is x. Let h be a node which is closest to u among the nodes on a path between s and t. Then there are two cases:  
1) h is on path between u and x

u

|

|

|

h x

/ \ /

/ \*

/ \

s t

then d(s, t) = d(s, h) + d(h, t) <= d(s, h) + d(h, x) = d(s, x), which contradicts assumption.

2) h is not on path between u and x

u

|

\*---x

|

h

/ \

/ \

/ \

s t

then

d(u, s) <= d(u, x) <= d(u, h) + d(h, x)  
d(u, t) <= d(u, x) <= d(u, h) + d(h, x)

d(s, t) = d(s, h) + d(h, t)  
= d(u, s) + d(u, t) - 2 d(u, h)  
<= 2 d(h, x)

2 d(s, t) <= d(s, t) + 2 d(h, x)  
= d(s, h) + d(h, x) + d(x, h) + d(h, t)  
= d(x, s) + d(x, t)

This means that max(d(v, s), d(v, t)) >= d(s, t), which also contradicts assumption.

Thus, we proved that farthest node of a node will be one of the endpoints of the longest path.

**Code:**

**Time Complexity**: O(N)

**Space Complexity:** O(height of tree)

**Approach two Solution:**

**Explanation:**

**Code:**

**Time Complexity**: O(N)

**Space Complexity:** O(height of tree)