Assignment 3 Theory Questions

Part 1 Theory Question (4 pts):

Q1.

Let T be a tree with n nodes. Define the lowest common ancestor (LCA) between two nodes v and w as the lowest node in T that has both v and w as descendants (where we allow a node to be adescendant of itself). Given two nodes v and w, describe an efficient algorithm for finding the LCA of v and w. What is the running time of your method?

To understand the problem, first we look at how to reach the Lowest Common Ancestor. We take the nodes v and w and check their depth which is done in lines 1 and 2 of the algorithm (basically the level at which they are). Then we need to make sure that they are at the same depth so we compare their depths and assign them their parent nodes until they have the same depth which is done in lines 4-6 and 7-9 of the algorithm. Now that they are at the same level, we compare their values and run a loop until they have the same while by assigning them their parent nodes in each iteration. This way we eventually reach a parent (basically an ancestor) which is common for both and is the Least Common Ancestor.

Algorithm:

```
1.  Function (Node v, Node w)
2.  DepthofV = v.depth
3.  DepthofW = w.depth
4.  While(DepthofV is greater than DepthofW)
5.   v = v.parentnode
6.  EndWhile
7.  While(DepthofW is greater than DepthofV)
8.   w = w.parentnode
9.  EndWhile
10. While(v is not equal to w)
11.  v = v.parentnode
12.  w = w.parentnode
13. Endwhile
14. Return v
15. End Function
```

Running Time:

The average running time for this algorithm should be the addition of all the loops. Therefore it will have an average runtime of 0(log v + log w) for execution. However, in the worst case which would be the case if there is only a single branch to the tree, meaning that the root only has the first child which inturn has only the first child and so on, ther running time will be 0(n) as the function will iterate through all the elements of the tree.

Q2.

Let T be a tree with n nodes and for any node v in T, let dv Denotes the depth of v in T. The distance between two nodes v and w in T is dv + dw − 2du, where node u is the LCA of v and w (as defined in the previous question Q1). The diameter of T is the maximum distance between two nodes in T. Describe an efficient algorithm for finding the diameter of T. What is the running time of your algorithm?

Here we need to get the diameter of the tree which is basically the maximum depth of the tree. In order to do that, we first make v as the deepest node in the tree through a function that we define in the class (line 2). In the case that the node v is the root, we return 0 (line 3). After this, we get the depth of this deepest node (line 8). Then we run a loop till v becomes the root as we assign it the parent node in each iteration (line 17). In each iteration, we assign the sibling of v to w (line 10) and then get the LCA for v and w and store it in X (line 11). Then we call a function to get the max depth of the sibling tree (the max depth of the descendants of w) and then store it in Y (line 12). Then we perform the calculation for the Distance between the nodes through the iteration using the formula given in the question (line 13). If this Distance in this iteration is greater than the previous iterations' Distance (if it is the maximum so far), we assign this value to the Diameter. (Line 14-16). This way we get the maximum diameter and then we return it (line 19).
Note: It goes on to the root as the sibling tree's depth is compared.

```
1.  Function diameter(node v, node w)
2.  v = deepestNode()
3.  If (v == TreeRoot())
4.  Return 0
5.  End If
6.  DTemp = 0
7.  Diameter = 0
8.  DV = v.depthOfNode()
9.  While(v is not equal to the root)
10.  w = v.Sibling()
11. X = LCA(v, w)
12. Y = w.maxDepth()
13. D = DV + Y - 2(X)
14. If(D>Diameter)
15. Diameter = D
16. EndIf
17.  v = v.parentnode
18. EndWhile
19. Return Diameter
20. End Function
```

Running Time:
The worst case running time for this algorithm should be 0(n) because in the worst case scenario the trees would have all the nodes in a single branch and the function would have to iterate through all the elements (nodes) in the tree.