Solutions for Assignment 3

Q1. It is simply a DFS, you need to do a DFS taking “source” as the root node of the tree and G as the tree suspended from Source.
You can modify TraverseFromVertex(G, u) as follows

```
1.
u.tree.depth=0
If d == 0:
        Return u
2.

11

        w.tree.depth=v.tree.depth+1
        If v.depth == d:
                store unique number on this node in any data structure
                *
12
         Return the lowest unique number from the data structure
```

Can you replace * with something that results in a better performance?

Q2. It is simply a DFS, but just gives up after visiting the first visited node. Hence, there is no way to visit more than |V| edges (the last edge should be a back edge), the complexity will be O(|V|).
In the first version of assignment, I asked for finding a loop instead of a cycle, which was a mistake. However, if you did that, it would be accepted.

Q3.
1. There are many solutions for this question, all using the following fact: if the DAG is semiconnected and $v_1 \ldots . v_n$ is its topological sorting, there is an edge between every v_i and v_{i+1}.

Proof: first we proof it is a necessary condition: for every v_i and v_{i+1} , there cannot be a path from v_{i+1} to v_i as it will form a loop. So the only possible way from v_i to v_{i+1} will be a direct edge. Second, we need to proof this condition is sufficient which is obvious, a path from v_i to v_{i+1} means that v_m will be connected to v_n for any m<n.

So the algorithm will be as simple as this: Get the topological order and check if all subsequent vertices are connected via a direct edge. The complexity is O(|V|+|E|)

2.I will leave the proof for the tutorial

1. Create all the SCC components
2. Create the **component graph** from it. Take a look at fig 22.9 of the textbook to see what a component graph is. But it simply means that condense a semi-directed components into one big node, and add edges between the condensed nodes A and B, if there is an edge like (a,b) where a is in A and b is in B.
3. The component graph is a DAG, so do the test that we did for part 1.

The complexity is O(|V|+|E|)